B. M.S. COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavangudi, Bangalore - 560019

MACHINE LEARNING LAB

(20CS6PCMAL)

Report

Submitted by

Lokesh R
(1BM20CS078)
VI B

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Lab Incharge

Prof Antara Roy Choudhury

Assistant Professor

BMS College Of Engineering

2023-2024

LIST OF PROGRAMS

Date: 01.04.2023

Program 1 – a) Dataset Exploration – Iris data set

- Features in the Iris dataset:
    1. sepal length in cm
    2. sepal width in cm
    3. petal length in cm
    4. petal width in cm
- Target classes to predict:
    1. Iris Setosa
    2. Iris Versicolour
    3. Iris Virginica

In [ ]:
```
from sklearn.datasets import load_iris
iris=load_iris()
```

In [ ]:
```
iris
```

Out[ ]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
```

In [ ]:
```
type(iris)
```

Out[ ]:
```
sklearn.utils._bunch.Bunch
```

In [ ]:
```
iris.keys()
```

Out[ ]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

In [ ]:
```
print(iris["target_names"])
```

```
['setosa' 'versicolor' 'virginica']
```

In [ ]:
```
n_samples,n_features=iris.data.shape
print("no of samples",n_samples)
print("no of features",n_features)
print(iris.data[0])
```

```
no of samples 150
no of features 4
[5.1 3.5 1.4 0.2]
```

In [ ]:
```
iris.data[[12,26,89,114]]
```

Out[ ]:
```
array([[4.8, 3. , 1.4, 0.1],
       [5. , 3.4, 1.6, 0.4],
       [5.5, 2.5, 4. , 1.3],
       [5.8, 2.8, 5.1, 2.4]])
```

In [ ]:
```
print(iris.data.shape)
print(iris.target.shape)
```

```
(150, 4)
(150,)
```

In [ ]:
```
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [ ]:
```
import numpy as np
np.bincount(iris.target)
```

Out[ ]:
```
array([50, 50, 50])
```

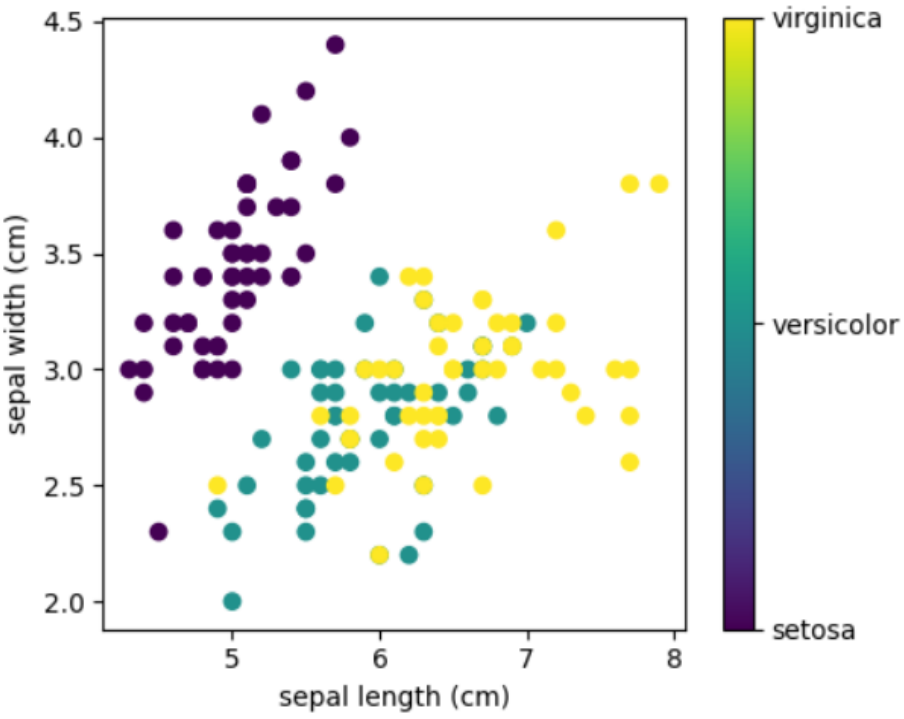
In [ ]:
```
import pandas as pd
iris=pd.DataFrame(data=np.c_[iris['data'],iris['target']],columns=iris['feature_names']+['target'])
iris.head(10)
```

Out[ ]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0
5	5.4	3.9	1.7	0.4	0.0
6	4.6	3.4	1.4	0.3	0.0
7	5.0	3.4	1.5	0.2	0.0
8	4.4	2.9	1.4	0.2	0.0
9	4.9	3.1	1.5	0.1	0.0

In [ ]:

```
from matplotlib import pyplot as plt


x_index = 0
y_index = 1


formatter = plt.FuncFormatter(lambda i, *args: iris.target_names[int(i)])

plt.figure(figsize=(5, 4))
plt.scatter(iris.data[:, x_index], iris.data[:, y_index], c=iris.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel(iris.feature_names[x_index])
plt.ylabel(iris.feature_names[y_index])

plt.tight_layout()
plt.show()
```



b. Practice Exercise - Dataset Exploration – Wine data set

In [ ]:
```
from sklearn.datasets import load_wine
wine=load_wine()
```

In [ ]:
```
wine.keys()
```

Out[ ]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names'])
```

In [ ]:
```
print(wine.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

    :Number of Instances: 178
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
                - Alcohol
                - Malic acid
                - Ash
                - Alcalinity of ash
                - Magnesium
                - Total phenols
                - Flavanoids
                - Nonflavanoid phenols
                - Proanthocyanins
                - Color intensity
                - Hue
                - OD280/OD315 of diluted wines
                - Proline

    - class:
            - class_0
            - class_1
            - class_2
```

In [ ]:
```
print(wine.target_names)
```

```
['class_0' 'class_1' 'class_2']
```

In [ ]:
```
print(wine.feature_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_int
ensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
```

In [ ]:
```
import numpy as np
np.bincount(wine.target)
```

Out[ ]:
```
array([59, 71, 48])
```

Date:8/4/2023

Program 2 – Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples .Read the training data from a .CSV file

Data set:Enjoysport

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Observation:

8/4/23

Lab 2 - Find S Algorithm

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

Data set: Enjoy sport

Example	Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Algorithm of Find-S

1) Initialize h to most specific hypothesis in H

2) For each positive training instance x

For each attribute constraint a_i in h

If the constraint a_i is satisfied by x

Then do nothing

else replace a_i in h by the next more general constraint that is

satisfied by x

3. Output hypothesis h

$h = \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$

$x_1 = \langle$ Sunny, Warm Normal Strong Warm Same $\rangle$

$h_1 = \langle$ Sunny Warm Normal Strong Warm Same $\rangle$

$x_2 = \langle$ Sunny Warm High Strong Warm Same $\rangle$

$h_2 = \langle$ Sunny Warm ? Strong Warm Same $\rangle$

$x_3 = \langle$ Rainy Cold High Strong Warm Change $\rangle$

$h_3 = \langle$ Sunny Warm ? Strong Warm Same $\rangle$

$x_4 = \langle$ Sunny Warm High Strong Cool Change $\rangle$

$h_4 = \langle$ Sunny Warm ? Strong ? ? $\rangle$

1. import numpy as np
   import pandas as pd

2. from google.colab import drive
   drive.mount("/content/drive")

3. path = "/content/drive/MyDrive/ML Lab/Lab 2/enjoysport.csv"

3. data_first = pd.read_csv(path)

4. print(data_first, "\n")

5. d = np.array(data_first)[:, :-1]
print("attributes are", d)

6. target = np.array(data_first)[:, -1]
print("target is", target)

target is ['Yes' 'Yes' 'NO' 'Yes']

7. def findS(c,t):
    for i,val in enumerate(t):
    if val == "Yes":
        specific_hypothesis = c[i].copy()
        break
    for i,val in enumerate(c):
    if t[i] == "Yes":
    for x in range(len(specific_hypothesis)):
        if val[x] != specific_hypothesis[x]:
        specific_hypothesis[x] = '?'
        else:
            pass
    return specific_hypothesis
print("final hypothesis is", findS(d,target))

final hypothesis is ['Sunny' 'Warm' '?' 'Strong' '?' '?']

The first step of **FIND-S** is to initialize h to the most specific hypothesis in H h — (Ø, Ø, Ø, Ø, Ø, Ø)

1. First training example **x1 = < Sunny, Warm, Normal, Strong ,Warm ,Same>,** EnjoySport = +ve. Observing the first training example, it is clear that hypothesis h is too specific. None of the "Ø" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example **h1 = < Sunny, Warm, Normal, Strong ,Warm, Same>.**

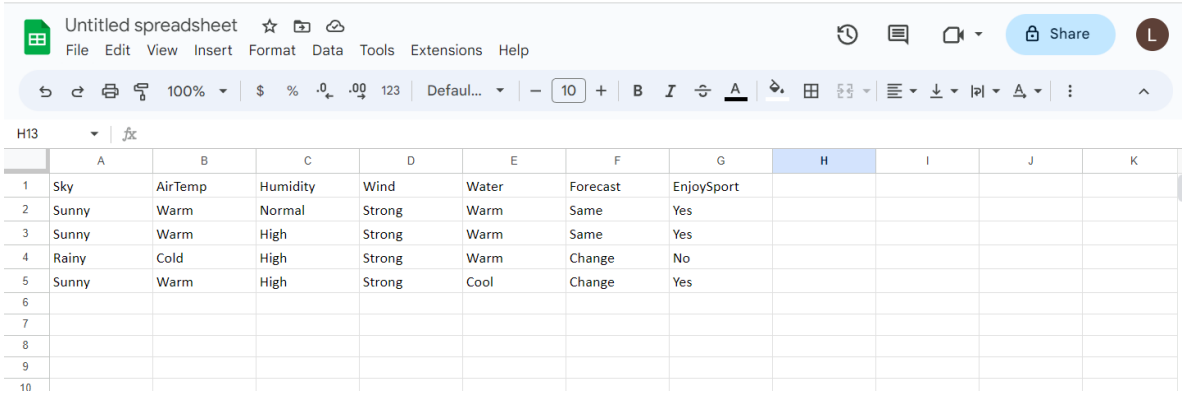
2. Consider the second training example **x2 = < Sunny, Warm, High, Strong, Warm, Same>**, EnjoySport = +ve. The second training example forces the algorithm to further generalize h, this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example. Now **h2 =< Sunny, Warm, ?, Strong, Warm, Same>**

3. Consider the third training example **x3 =< Rainy, Cold, High, Strong, Warm, Change>,**EnjoySport = — ve. The FIND-S algorithm simply ignores every negative example. So the hypothesis remain as before, so **h3 =< Sunny, Warm, ?, Strong, Warm, Same>**

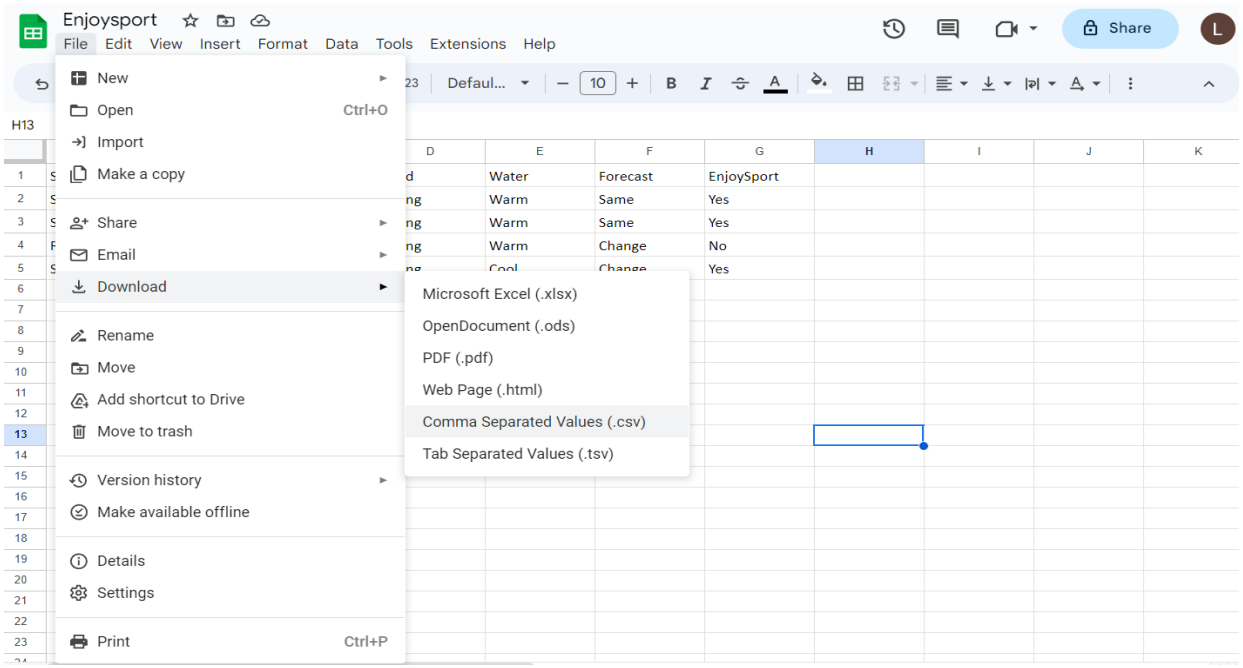
4. Consider the fourth training example **x4 =<Sunny,Warm ,High, Strong, Cool, Change>**, EnjoySport =+ve. The fourth example leads to a further generalization of h as **h4 =< Sunny, Warm, ?, Strong, ?, ?>**

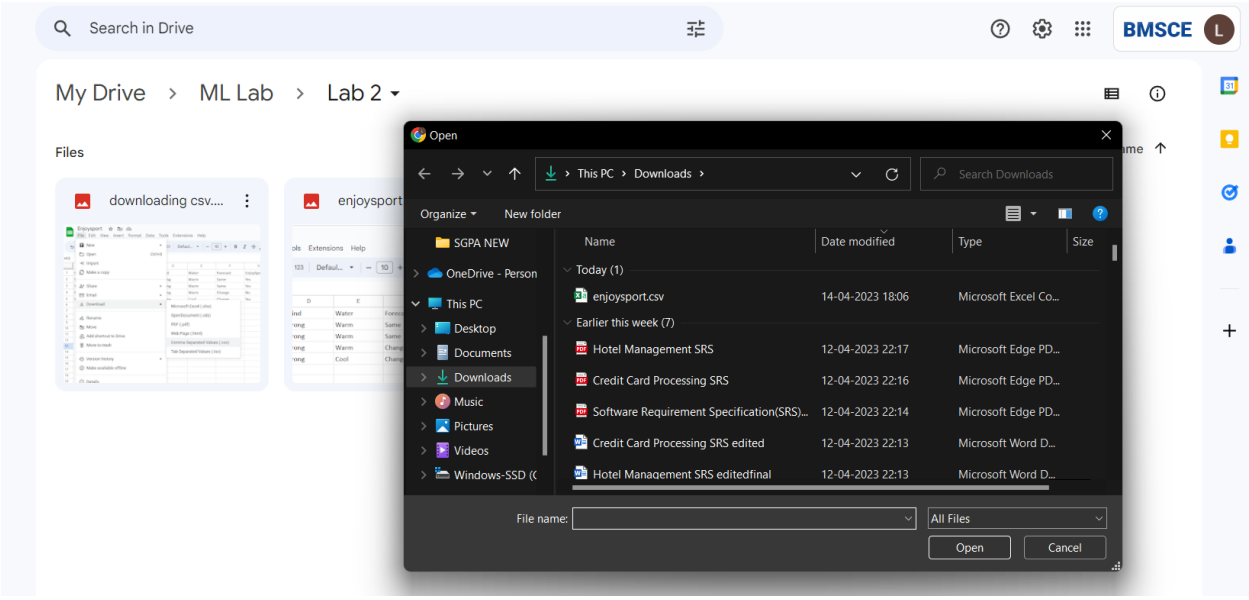
5. So the final hypothesis is **< Sunny, Warm, ?, Strong, ?, ?>**

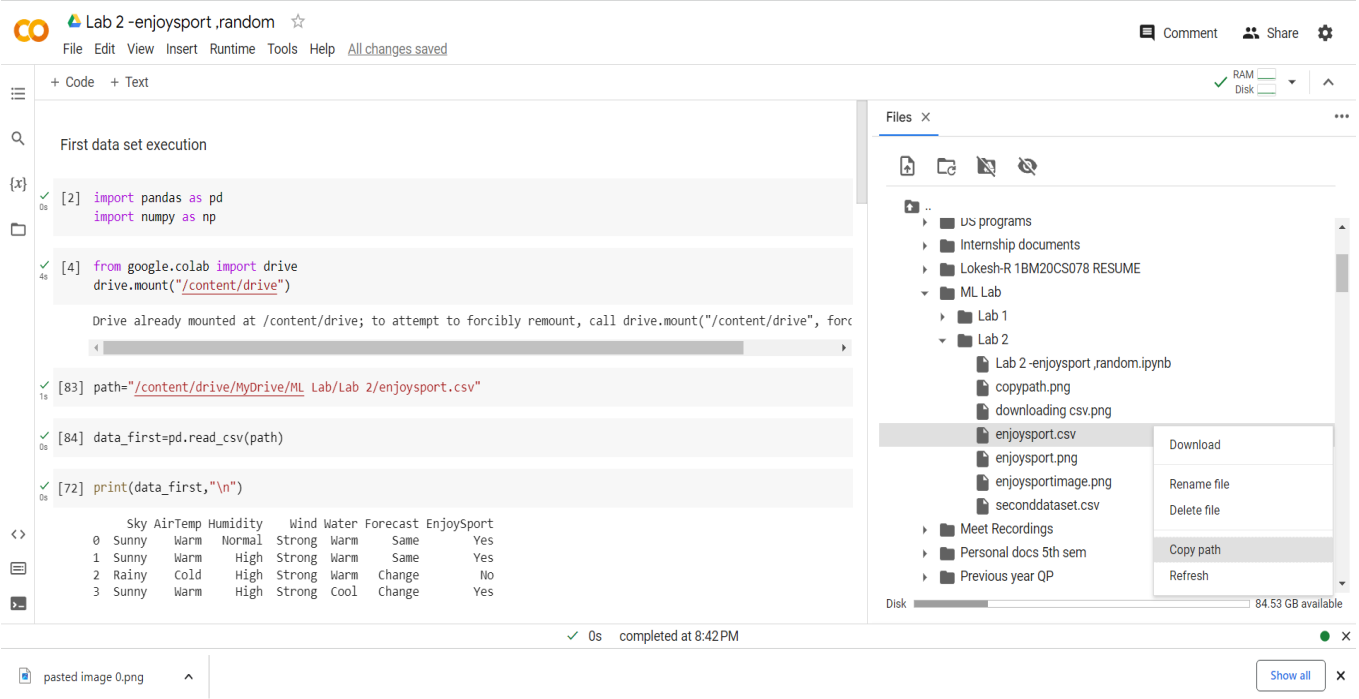
CSV file creation :



	A	B	C	D	E	F	G
1	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
2	Sunny	Warm	Normal	Strong	Warm	Same	Yes
3	Sunny	Warm	High	Strong	Warm	Same	Yes
4	Rainy	Cold	High	Strong	Warm	Change	No
5	Sunny	Warm	High	Strong	Cool	Change	Yes




Search in Drive
BMSCE
My Drive > ML Lab > Lab 2
Files
downloading csv....
enjoysport
Open
This PC > Downloads >
Search Downloads
Organize
New folder
SGPA NEW
OneDrive - Person
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD
Name
Date modified
Type
Size
Today (1)
enjoysport.csv 14-04-2023 18:06 Microsoft Excel Co...
Earlier this week (7)
Hotel Management SRS 12-04-2023 22:17 Microsoft Edge PD...
Credit Card Processing SRS 12-04-2023 22:16 Microsoft Edge PD...
Software Requirement Specification(SRS)... 12-04-2023 22:14 Microsoft Edge PD...
Credit Card Processing SRS edited 12-04-2023 22:13 Microsoft Word D...
Hotel Management SRS editedfinal 12-04-2023 22:13 Microsoft Word D...
File name:
All Files
Open
Cancel


Lab 2 -enjoysport ,random
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
First data set execution
[2] import pandas as pd
import numpy as np
[4] from google.colab import drive
drive.mount("/content/drive")
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
[83] path="/content/drive/MyDrive/ML Lab/Lab 2/enjoysport.csv"
[84] data_first=pd.read_csv(path)
[72] print(data_first,"\n")
Sky AirTemp Humidity Wind Water Forecast EnjoySport
0 Sunny Warm Normal Strong Warm Same Yes
1 Sunny Warm High Strong Warm Same Yes
2 Rainy Cold High Strong Warm Change No
3 Sunny Warm High Strong Cool Change Yes
Files
DS programs
Internship documents
Lokesh-R 1BM20CS078 RESUME
ML Lab
Lab 1
Lab 2
Lab 2 -enjoysport ,random.ipynb
copypath.png
downloading csv.png
enjoysport.csv
enjoysport.png
enjoysportimage.png
seconddataset.csv
Meet Recordings
Personal docs 5th sem
Previous year QP
Download
Rename file
Delete file
Copy path
Refresh
Disk
84.53 GB available
0s completed at 8:42 PM
pasted image 0.png
Show all

Program 1:

Open in Colab

First data set execution

In [2]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [83]:
```
path="/content/drive/MyDrive/ML Lab/Lab 2/enjoysport.csv"
```

In [84]:
```
data_first=pd.read_csv(path)
```

In [72]:
```
print(data_first,"\n")
```

```
     Sky AirTemp Humidity    Wind Water Forecast EnjoySport
0  Sunny    Warm   Normal  Strong  Warm     Same        Yes
1  Sunny    Warm     High  Strong  Warm     Same        Yes
2  Rainy    Cold     High  Strong  Warm   Change         No
3  Sunny    Warm     High  Strong  Cool   Change        Yes
```

In [85]:
```
d=np.array(data_first)[:,:-1]
print("attributes are:",d)
```

```
attributes are: [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

In [86]:
```
target=np.array(data_first)[:,-1]
print("target is:",target)
```

```
target is: ['Yes' 'Yes' 'No' 'Yes']
```

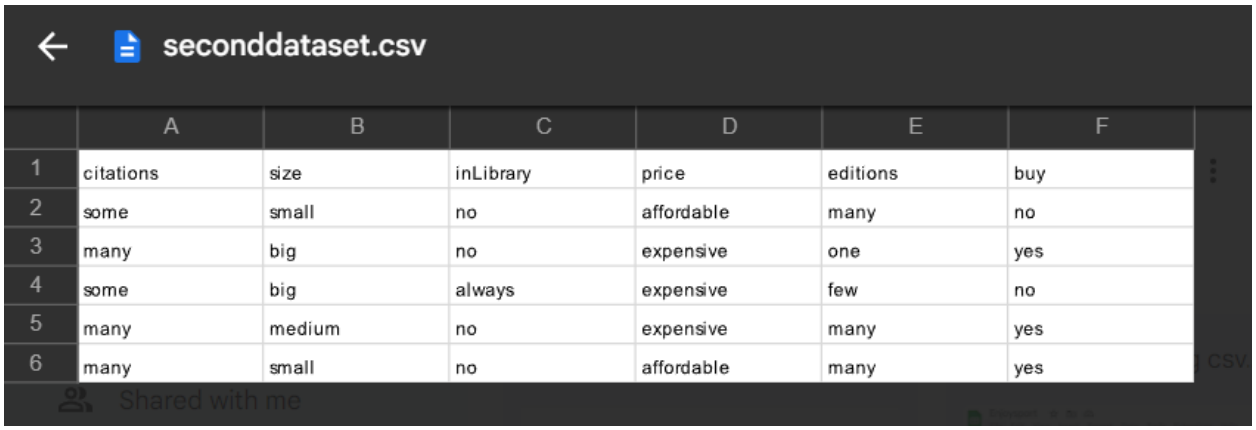
In [87]:
```
def findS(c,t):
    for i,val in enumerate(t):
      if val=="Yes":
        specific_hypothesis=c[i].copy()
        break
    for i,val in enumerate(c):
      if t[i]=="Yes":
        for x in range (len(specific_hypothesis)):
          if val[x]!=specific_hypothesis[x]:
            specific_hypothesis[x]='?'
          else:
              pass
    return specific_hypothesis
print("final hypothesis is",findS(d,target))
```

```
final hypothesis is ['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

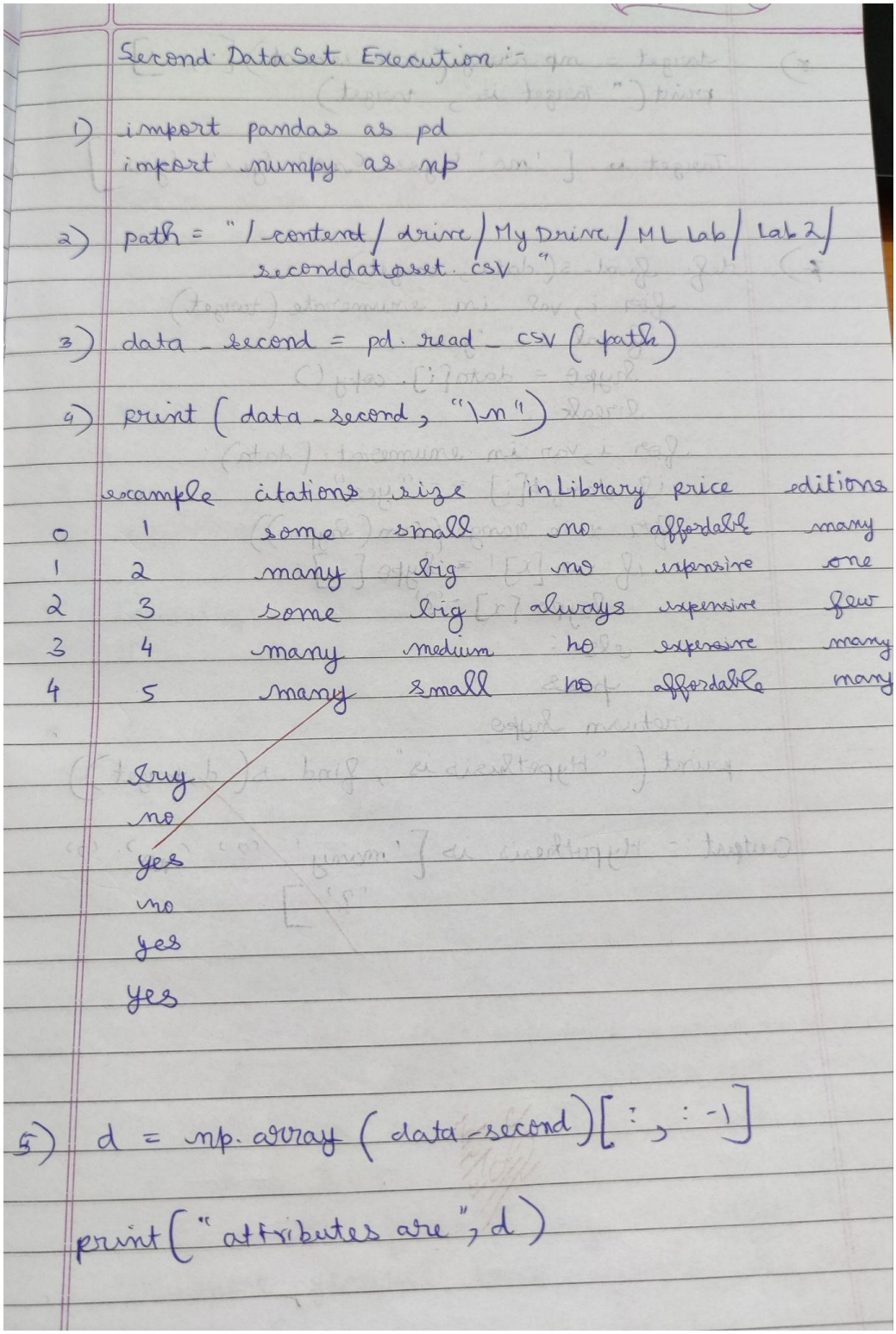
Second dataset-

example	*citations*	*size*	*inLibrary*	*price*	*editions*	*buy*
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

CSV file contents:



Observation :



Second Dataset Execution :

1) import pandas as pd
import numpy as np

2) path = "/content/drive/My Drive/ML Lab/Lab 2/seconddataset.csv"

3) data_second = pd.read_csv(path)

4) print(data_second, "\n")

	example	citations	size	inLibrary	price	editions
0	1	some	small	no	affordable	many
1	2	many	big	no	expensive	one
2	3	some	big	always	expensive	few
3	4	many	medium	no	expensive	many
4	5	many	small	no	affordable	many

Buy
no
yes
no
yes
yes

5) d = np.array(data_second)[:, :-1]

print("attributes are", d)

```
target = np.array(data.second)[:, -1]
print("Target is", target)
```

Target is ['no' 'yes' 'no' 'yes' 'yes']

```
def find_s(data, target)
    for i, val in enumerate(target):
        if val == 'yes':
            hypo = data[i].copy()
            break
    for i, var in enumerate(data):
        if target[i] == "yes":
            for x in range(len(hypo)):
                if var[x] != hypo[x]:
                    hypo[x] = '?'
                else:
                    pass
    return hypo
print("Hypothesis is", find_s(d, target))
```

Output :- Hypothesis is ['many' '?' 'no' '?' '?']

Program 2 execution:

Second dataset execution-

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
path="/content/drive/MyDrive/ML Lab/Lab 2/seconddataset.csv"
```

In [3]:
```
data_second=pd.read_csv(path)
```

In [4]:
```
print(data_second,"\n")
```

```
  citations    size inLibrary       price editions  buy
0      some   small        no  affordable     many   no
1      many     big        no   expensive      one  yes
2      some     big    always   expensive      few   no
3      many  medium        no   expensive     many  yes
4      many   small        no  affordable     many  yes
```

In [5]:
```
d=np.array(data_second)[:,:-1]
print("attributes are:",d)
```

```
attributes are: [['some' 'small' 'no' 'affordable' 'many']
 ['many' 'big' 'no' 'expensive' 'one']
 ['some' 'big' 'always' 'expensive' 'few']
 ['many' 'medium' 'no' 'expensive' 'many']
 ['many' 'small' 'no' 'affordable' 'many']]
```

In [6]:
```
target=np.array(data_second)[:,-1]
print("target is:",target)
```

```
target is: ['no' 'yes' 'no' 'yes' 'yes']
```

In [7]:
```
def find_s(data,target):
  for i,val in enumerate(target):
    if val=='yes':
      hypo=data[i].copy()
      break

  for i,var in enumerate(data):
    if target[i]=="yes":
      for x in range(len(hypo)):
        if var[x]!=hypo[x]:
          hypo[x]='?'
        else:
          pass

  return hypo

print("The Hypothesis is",find_s(d,target))
```

```
The Hypothesis is ['many' '?' 'no' '?' '?']
```

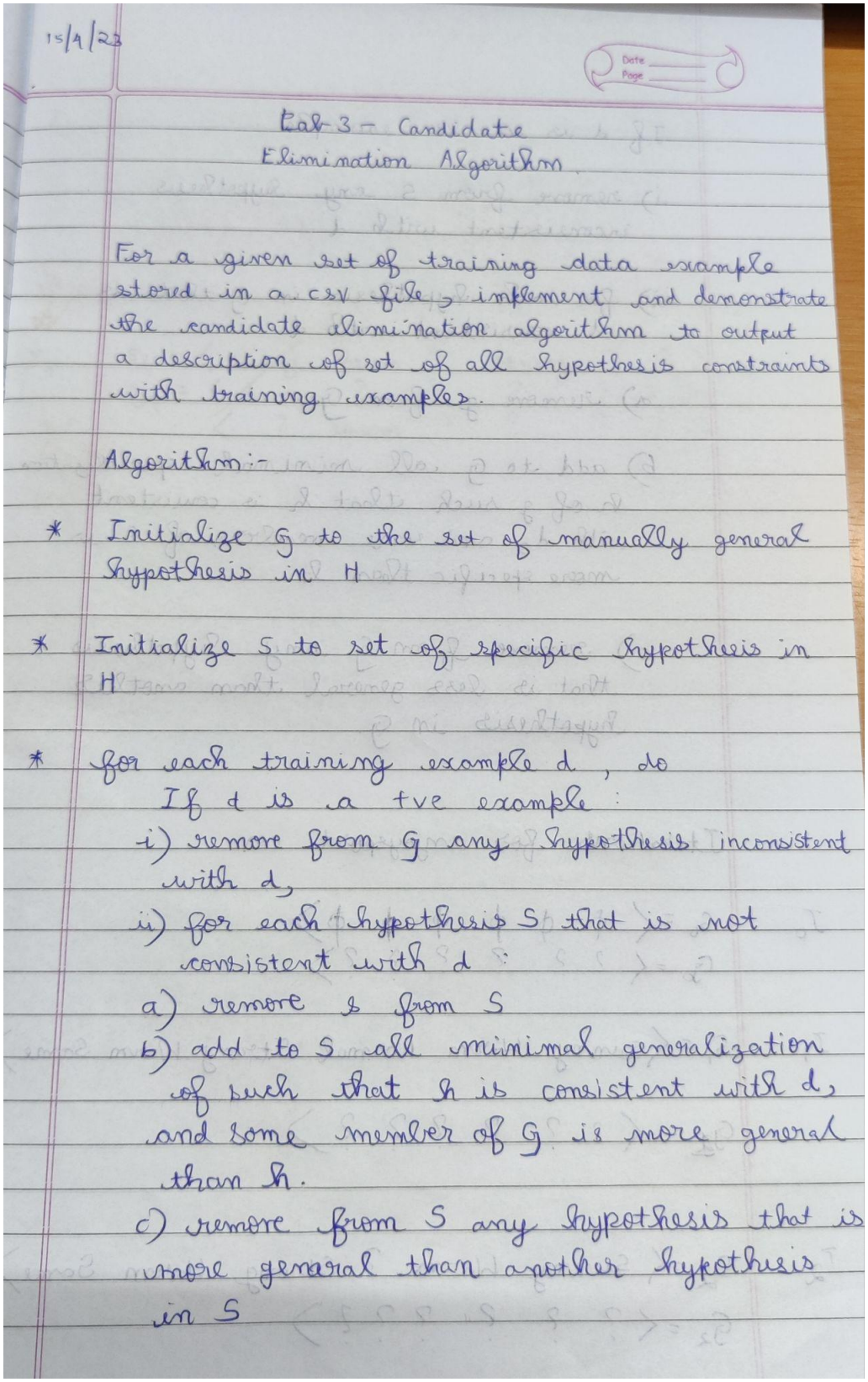
Date:15/4/23

Program 3- For a given set of training data examples stored in a .csv file ,implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypothesis consistent with training examples.

Data set-EnjoySport

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Observation :



15/9/23

Lab 3 – Candidate Elimination Algorithm

For a given set of training data example stored in a .csv file, implement and demonstrate the candidate elimination algorithm to output a description of set of all hypotheses constraints with training examples.

Algorithm :-

* Initialize G to the set of manually general hypotheses in H
* Initialize S to set of specific hypotheses in H
* for each training example d, do
  If d is a +ve example :
  i) remove from G any hypothesis inconsistent with d,
  ii) for each hypothesis S that is not consistent with d :
  a) remove s from S
  b) add to S all minimal generalization of such that h is consistent with d, and some member of G is more general than h.
  c) remove from S any hypothesis that is more general than another hypothesis in S

If d is a +ve example:

i) remove from S any hypothesis inconsistent with d

ii) for each hypothesis in G that is not consistent with d

a) remove g from G

b) add to G all minimal specialization h of g such that h is consistent with d_2 and some members of S is more specific than h.

c) remove from G any hypothesis that is less general than another hypothesis in G.

Iterations for enjoysport.csv

I_0 $S_0 = \langle \phi \ \phi \ \phi \ \phi \ \phi \ \phi \rangle$

$G_0 = \langle ? \ ? \ ? \ ? \ ? \ ? \rangle$

I_1 $S_1 = \langle$ Sunny Warm Normal Strong Warm Same $\rangle$

$G_1 = \langle ? \ ? \ ? \ ? \ ? \ ? \rangle$

I_2 $S_2 = \langle$ Sunny Warm ? Strong Warm Same $\rangle$

$G_2 = \langle ? \ ? \ ? \ ? \ ? \ ? \rangle$

I_3 $S_3 = \langle$ Sunny Warm ? Strong Warm Same $\rangle$

$G_3 = \langle$ Sunny ? ? ? ? ? $\rangle$

$\langle$? Warm ? ? ? ? $\rangle$

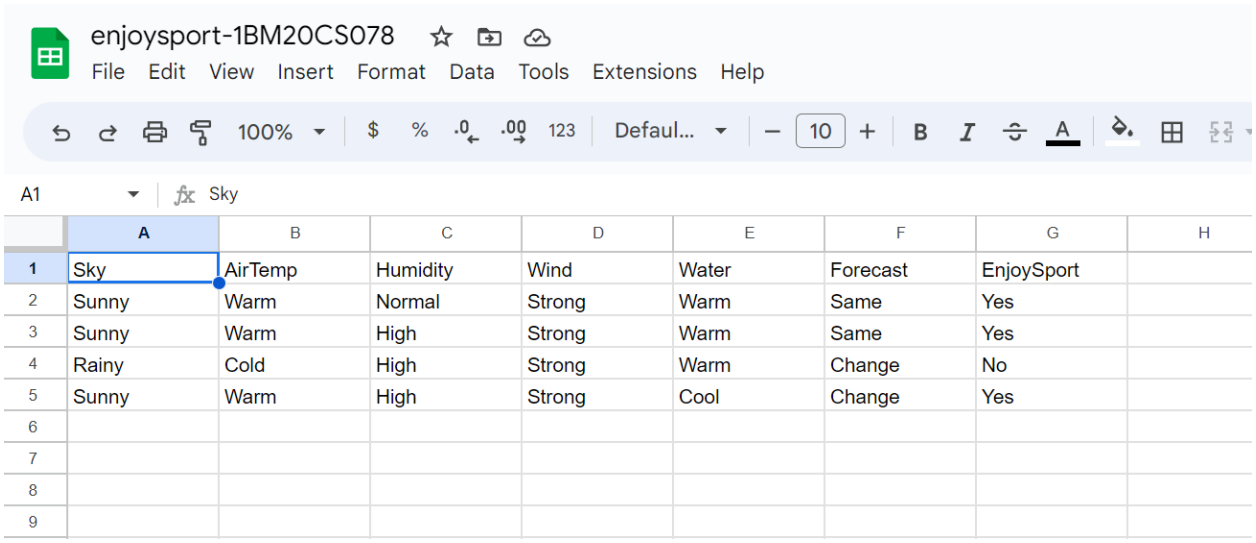
$\langle$? ? ? ? ? Same $\rangle$

I_4 $S_4 = \langle$ Sunny Warm ? Strong ? ? $\rangle$

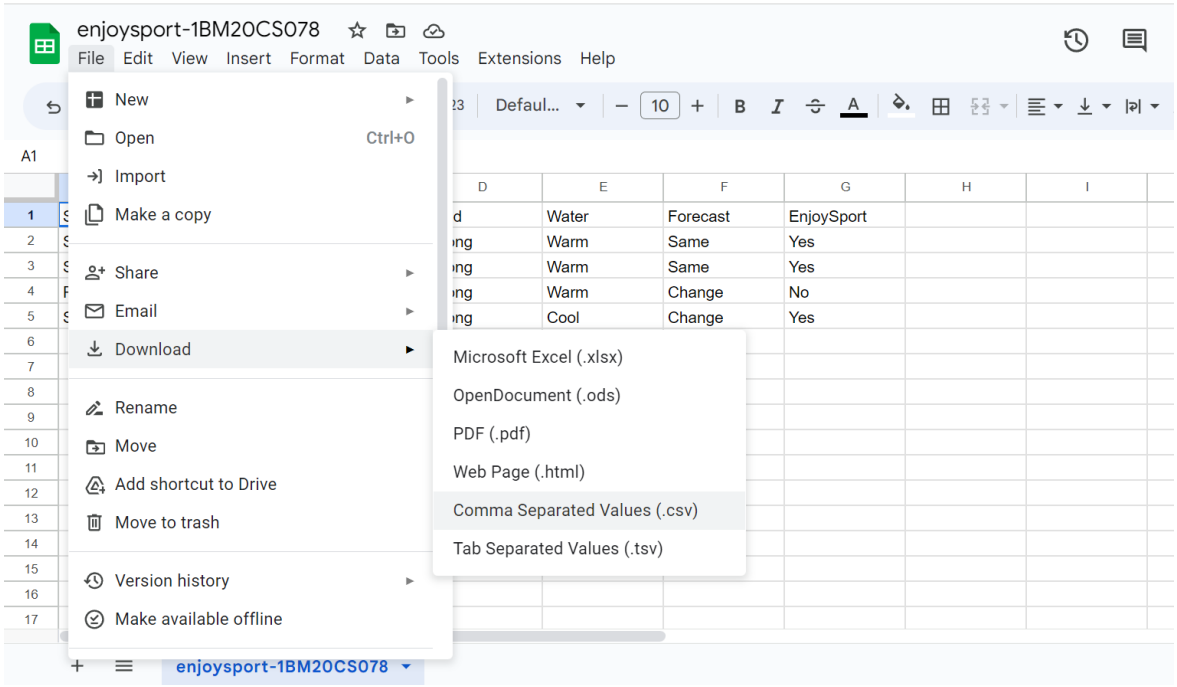
$G_4 = \langle$ Sunny ? ? ? ? ? $\rangle$

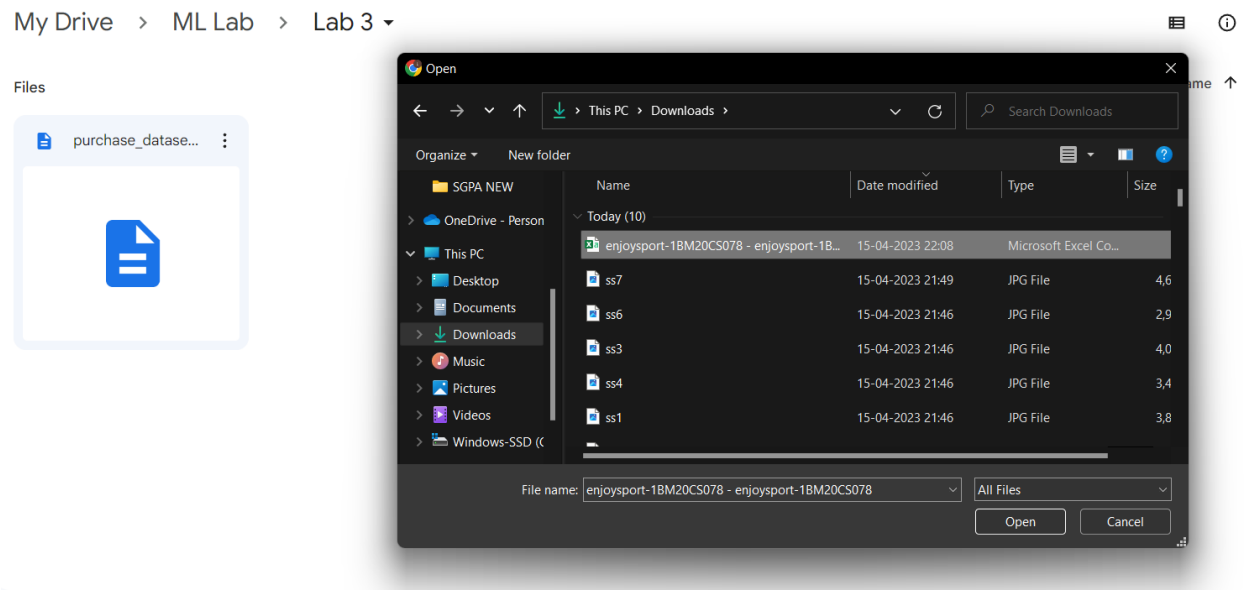
$\langle$? Warm ? ? ? ? $\rangle$

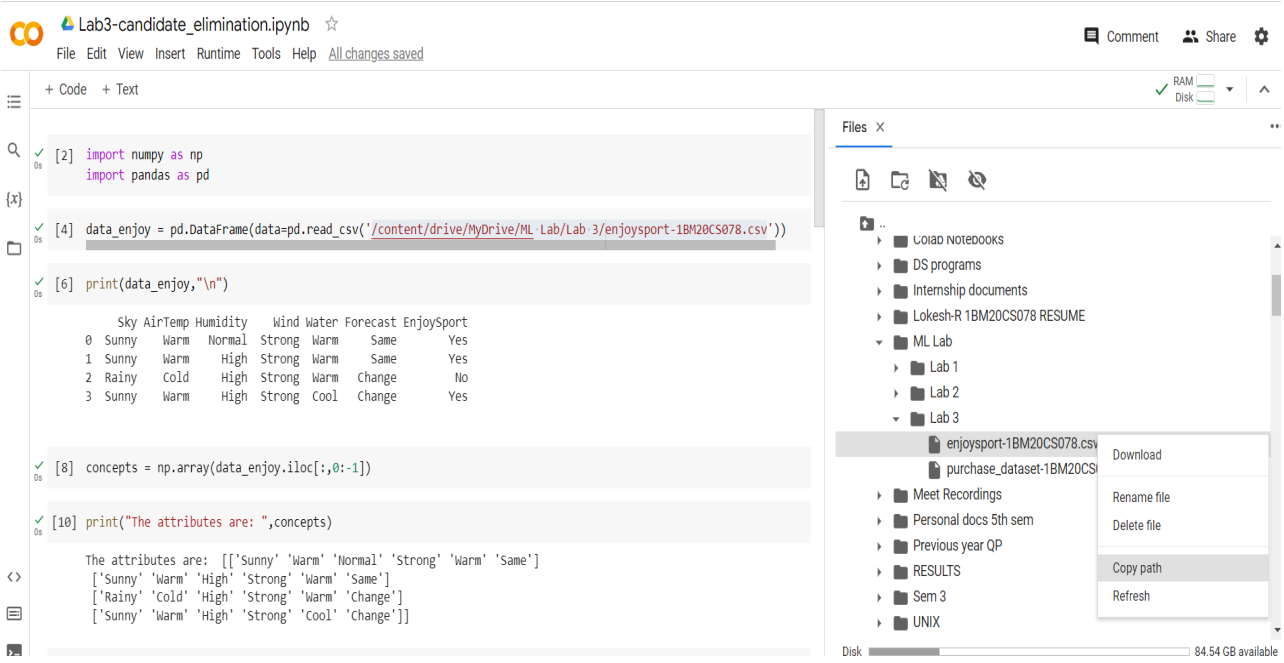
Uploading csv :



	A	B	C	D	E	F	G	H
1	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport	
2	Sunny	Warm	Normal	Strong	Warm	Same	Yes	
3	Sunny	Warm	High	Strong	Warm	Same	Yes	
4	Rainy	Cold	High	Strong	Warm	Change	No	
5	Sunny	Warm	High	Strong	Cool	Change	Yes	
6								
7								
8								
9								




My Drive > ML Lab > Lab 3
Files
purchase_datase...
Open
This PC > Downloads >
Search Downloads
Organize
New folder
SGPA NEW
OneDrive - Person
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows-SSD (C
Name
Date modified
Type
Size
Today (10)
enjoysport-1BM20CS078 - enjoysport-1B... 15-04-2023 22:08 Microsoft Excel Co...
ss7 15-04-2023 21:49 JPG File 4,6
ss6 15-04-2023 21:46 JPG File 2,9
ss3 15-04-2023 21:46 JPG File 4,0
ss4 15-04-2023 21:46 JPG File 3,4
ss1 15-04-2023 21:46 JPG File 3,8
File name: enjoysport-1BM20CS078 - enjoysport-1BM20CS078
All Files
Open
Cancel


Lab3-candidate_elimination.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[2] import numpy as np
import pandas as pd
[4] data_enjoy = pd.DataFrame(data=pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 3/enjoysport-1BM20CS078.csv'))
[6] print(data_enjoy,"\n")
Sky AirTemp Humidity Wind Water Forecast EnjoySport
0 Sunny Warm Normal Strong Warm Same Yes
1 Sunny Warm High Strong Warm Same Yes
2 Rainy Cold High Strong Warm Change No
3 Sunny Warm High Strong Cool Change Yes
[8] concepts = np.array(data_enjoy.iloc[:,0:-1])
[10] print("The attributes are: ",concepts)
The attributes are: [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
Files
Colab Notebooks
DS programs
Internship documents
Lokesh-R 1BM20CS078 RESUME
ML Lab
Lab 1
Lab 2
Lab 3
enjoysport-1BM20CS078.csv
purchase_dataset-1BM20CS
Meet Recordings
Personal docs 5th sem
Previous year QP
RESULTS
Sem 3
UNIX
Download
Rename file
Delete file
Copy path
Refresh
Disk
84.54 GB available

Program execution -

Open in Colab

In [2]:
```
import numpy as np
import pandas as pd
```

In [4]:
```
data_enjoy = pd.DataFrame(data=pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 3/enjoysport-1BM20CS078.csv'))
```

In [6]:
```
print(data_enjoy,"\n")
```

```
     Sky AirTemp Humidity    Wind Water Forecast EnjoySport
0  Sunny    Warm   Normal  Strong  Warm     Same        Yes
1  Sunny    Warm     High  Strong  Warm     Same        Yes
2  Rainy    Cold     High  Strong  Warm   Change         No
3  Sunny    Warm     High  Strong  Cool   Change        Yes
```

In [8]:
```
concepts = np.array(data_enjoy.iloc[:,0:-1])
```

In [10]:
```
print("The attributes are: ",concepts)
```

```
The attributes are:  [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

In [12]:
```
target = np.array(data_enjoy.iloc[:,-1])
```

In [14]:
```
print("\n The target is: ",target)
```

```
The target is:  ['Yes' 'Yes' 'No' 'Yes']
```

In [16]:
```
import csv
```

In [18]:
```
with open("/content/drive/MyDrive/ML Lab/Lab 3/enjoysport-1BM20CS078.csv") as t:
    csv_file = csv.reader(t)
    data = list(csv_file)

    specific = data[1][:-1]
    general = [['?' for i in range(len(specific))] for j in range(len(specific))]

    for i in data:
        if i[-1] == "Yes":
            for j in range(len(specific)):
                if i[j] != specific[j]:
                    specific[j] = "?"
                    general[j][j] = "?"

        elif i[-1] == "No":
            for j in range(len(specific)):
                if i[j] != specific[j]:
                    general[j][j] = specific[j]
                else:
                    general[j][j] = "?"

        print("\nStep " + str(data.index(i)) + " of Candidate Elimination Algorithm")
        print(specific)
        print(general)

    gh = [] # gh = general Hypothesis
    for i in general:
        for j in i:
            if j != '?':
                gh.append(i)
                break
    print("\nFinal Specific hypothesis:\n", specific)
    print("\nFinal General hypothesis:\n", gh)
```

```
Step 0 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 1 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 2 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 3 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]

Step 4 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', '?', '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific hypothesis:
 ['Sunny', 'Warm', '?', 'Strong', '?', '?']

Final General hypothesis:
 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

Second dataset -

example	*citations*	*size*	*inLibrary*	*price*	*editions*	*buy*
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

CSV file

purchase_dataset-1BM20CS078.csv

	A	B	C	D	E	F
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

Program execution -

Second dataset

In [20]:
```
import pandas as pd
import numpy as np
```

In [22]:
```
data_purchase = pd.DataFrame(data=pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 3/purchase_dataset-1BM20CS078.csv'))
```

In [24]:
```
print(data_purchase,"\n")
```

```
   some   small      no  affordable  many no.1
0  many     big      no   expensive   one  yes
1  some     big  always   expensive   few   no
2  many  medium      no   expensive  many  yes
3  many   small      no  affordable  many  yes
```

In [26]:
```
concepts = np.array(data_purchase.iloc[:,0:-1])
```

In [28]:
```
print("The attributes are: ",concepts)
```

```
The attributes are:  [['many' 'big' 'no' 'expensive' 'one']
 ['some' 'big' 'always' 'expensive' 'few']
 ['many' 'medium' 'no' 'expensive' 'many']
 ['many' 'small' 'no' 'affordable' 'many']]
```

In [30]:
```
target = np.array(data_purchase.iloc[:,-1])
```

In [32]:
```
print("\n The target is: ",target)
```

```
 The target is:  ['yes' 'no' 'yes' 'yes']
```

Second dataset

In [20]:
```
import pandas as pd
import numpy as np
```

In [22]:
```
data_purchase = pd.DataFrame(data=pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 3/purchase_dataset-1BM20CS078.csv'))
```

In [24]:
```
print(data_purchase,"\n")
```

```
   some   small      no  affordable  many no.1
0  many     big      no   expensive   one  yes
1  some     big  always   expensive   few   no
2  many  medium      no   expensive  many  yes
3  many   small      no  affordable  many  yes
```

In [26]:
```
concepts = np.array(data_purchase.iloc[:,0:-1])
```

In [28]:
```
print("The attributes are: ",concepts)
```

```
The attributes are:  [['many' 'big' 'no' 'expensive' 'one']
 ['some' 'big' 'always' 'expensive' 'few']
 ['many' 'medium' 'no' 'expensive' 'many']
 ['many' 'small' 'no' 'affordable' 'many']]
```

In [30]:
```
target = np.array(data_purchase.iloc[:,-1])
```

In [32]:
```
print("\n The target is: ",target)
```

```
 The target is:  ['yes' 'no' 'yes' 'yes']
```

```
Step 0 of Candidate Elimination Algorithm
['many', 'big', 'no', 'expensive', 'one']
[['many', '?', '?', '?', '?'], ['?', 'big', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', 'expensive', '?'], ['?', '?', '?', '?', 'one']]

Step 1 of Candidate Elimination Algorithm
['many', 'big', 'no', 'expensive', 'one']
[['many', '?', '?', '?', '?'], ['?', 'big', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', 'expensive', '?'], ['?', '?', '?', '?', 'one']]

Step 2 of Candidate Elimination Algorithm
['many', 'big', 'no', 'expensive', 'one']
[['many', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'no', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', 'one']]

Step 3 of Candidate Elimination Algorithm
['many', '?', 'no', 'expensive', '?']
[['many', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'no', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?']]

Step 4 of Candidate Elimination Algorithm
['many', '?', 'no', '?', '?']
[['many', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'no', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?']]

Final Specific hypothesis:
 ['many', '?', 'no', '?', '?']

Final General hypothesis:
 [['many', '?', '?', '?', '?'], ['?', '?', 'no', '?', '?']]
```

Date:29-04-2023

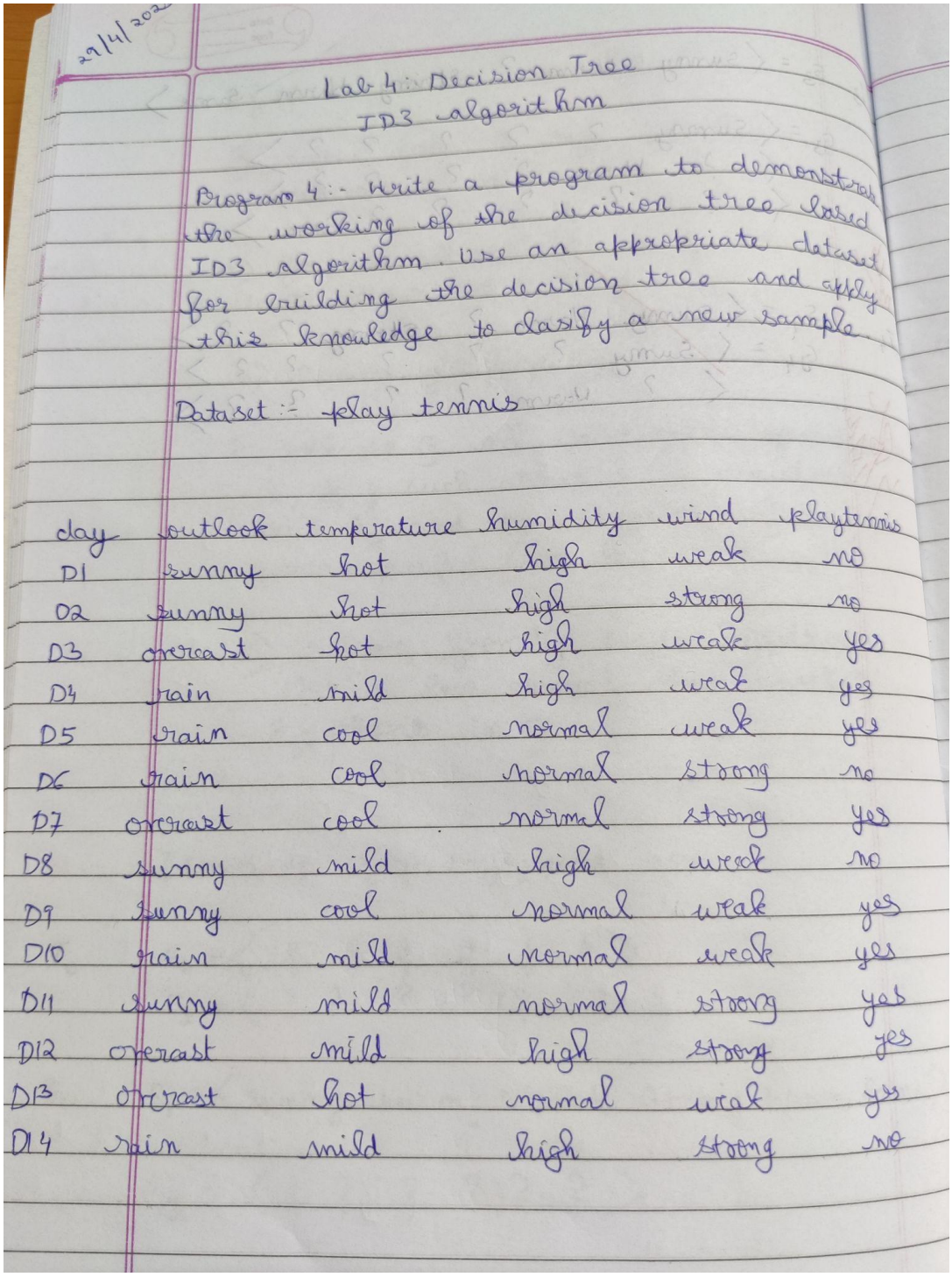
Program-4:

Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Dataset: Playtennis

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Observation -



29/4/202

Lab 4: Decision Tree
ID3 algorithm

Program 4:- Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate dataset for building the decision tree and apply this knowledge to classify a new sample.

Dataset :- play tennis

day	outlook	temperature	humidity	wind	playtennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

Algorithm

ID3 (examples, target - attribute, attributes)

Examples are the training examples, target - attributes is the attribute whose value is to be predicted by the tree. Attribute is the list of other attributes that may be treated by the learned decision tree. Returns a decision tree that correctly classifies the given examples

→ Create a root node of the tree.

→ If all examples are positive, return the single node tree root with label = +

→ If all examples are negative, return the single node tree root with label = -

→ If attribute is empty, return the single node tree root, with label = most common value of target attribute

→ otherwise begin

i) A ← The attribute from attribute that best classifies examples.

ii) The decision attributes for root ← A

iii) For each possible value, V_i of A

Add a new tree branch below root, corresponding to test $A = V_i$

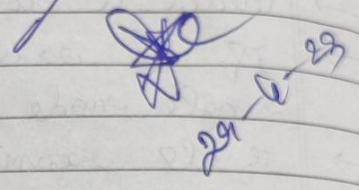
iv) Let examples V_i, be the subset of examples that have a value V_i for A

v) If examples v_i is empty

Then below this new branch add a leaf node with label = most common attribute of target - attribute in examples

else

below this new branch add the subtree ID3 (examples v_i, Target - attributes, attributes)



Output:-

```
Outlook
  rain
    wind
      strong
        no
      weak
        yes
  sunny
    humidity
      normal
        yes
      high
        no
  overcast
    yes
```

Entropy(S) = $-P_{\oplus} \log P_{+} - P_{\ominus} \log_2 P_{\ominus}$

$= \frac{-9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14}$

$= 0.94$

Entropy(S_{sunny}) = $\frac{-2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$

Entropy($S_{overcast}$) = $\frac{-4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$

Entropy(S_{rain}) = $\frac{-3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$

Gain(S, outlook) = $S - \sum_{v \in values} \frac{|S_v|}{|S|}$ entropy(S_v)

$= E(S) - \frac{5}{14} E(S_{sunny}) - \frac{4}{14} E(S_{overcast}) - \frac{5}{14} E(S_{rain})$

$= 0.2462.$

$E(S_{hot}) = 1$
$E(S_{cold}) = 0.9183$
$E(S_{mild}) = 0.8113$

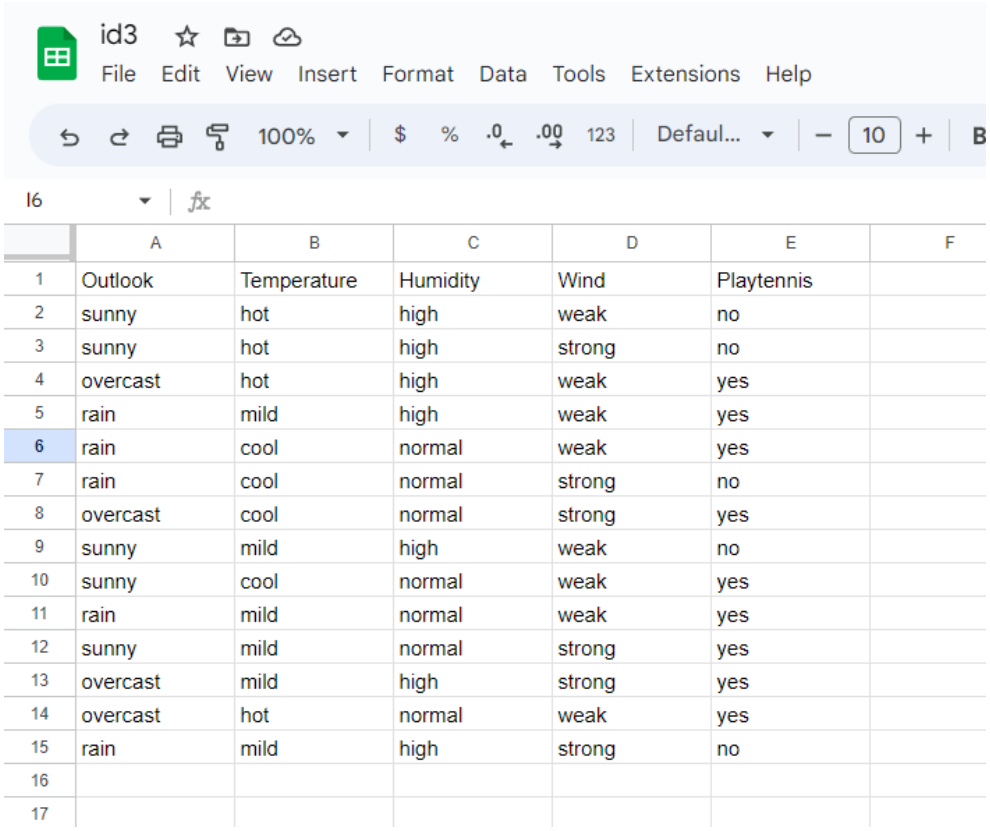
$G(S, temp) = 0.0289$

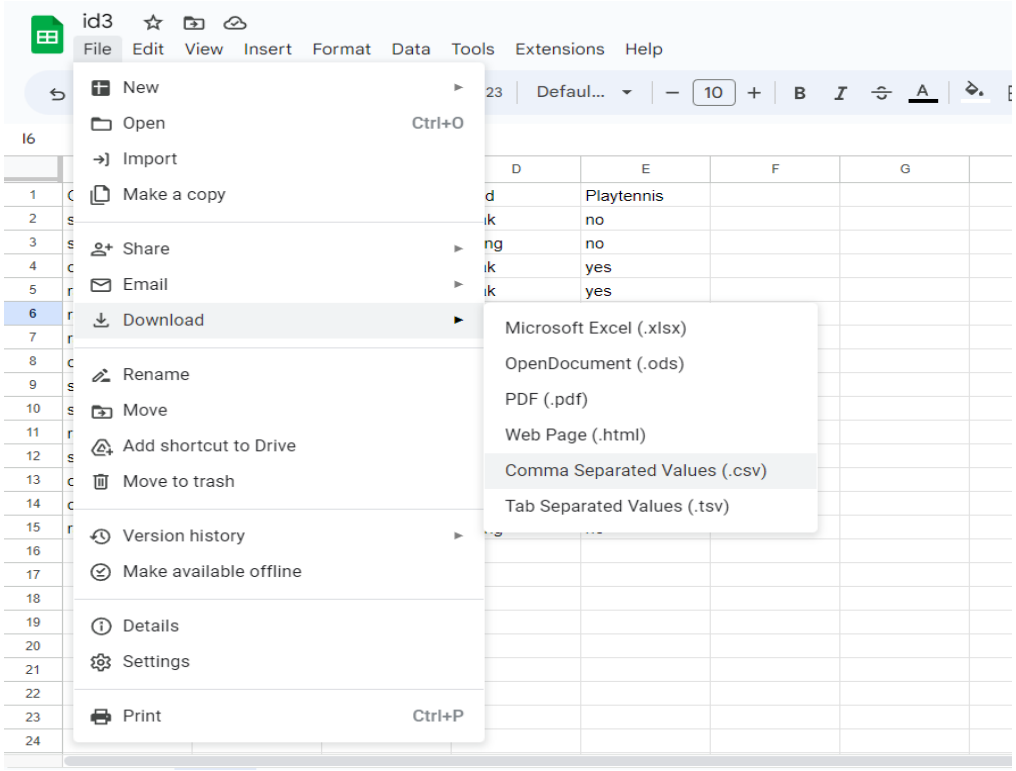
$E(S_{high}) = 0.9852$
$E(S_{normal}) = 0.5916$
$G(S, humidity) = 0.1516$

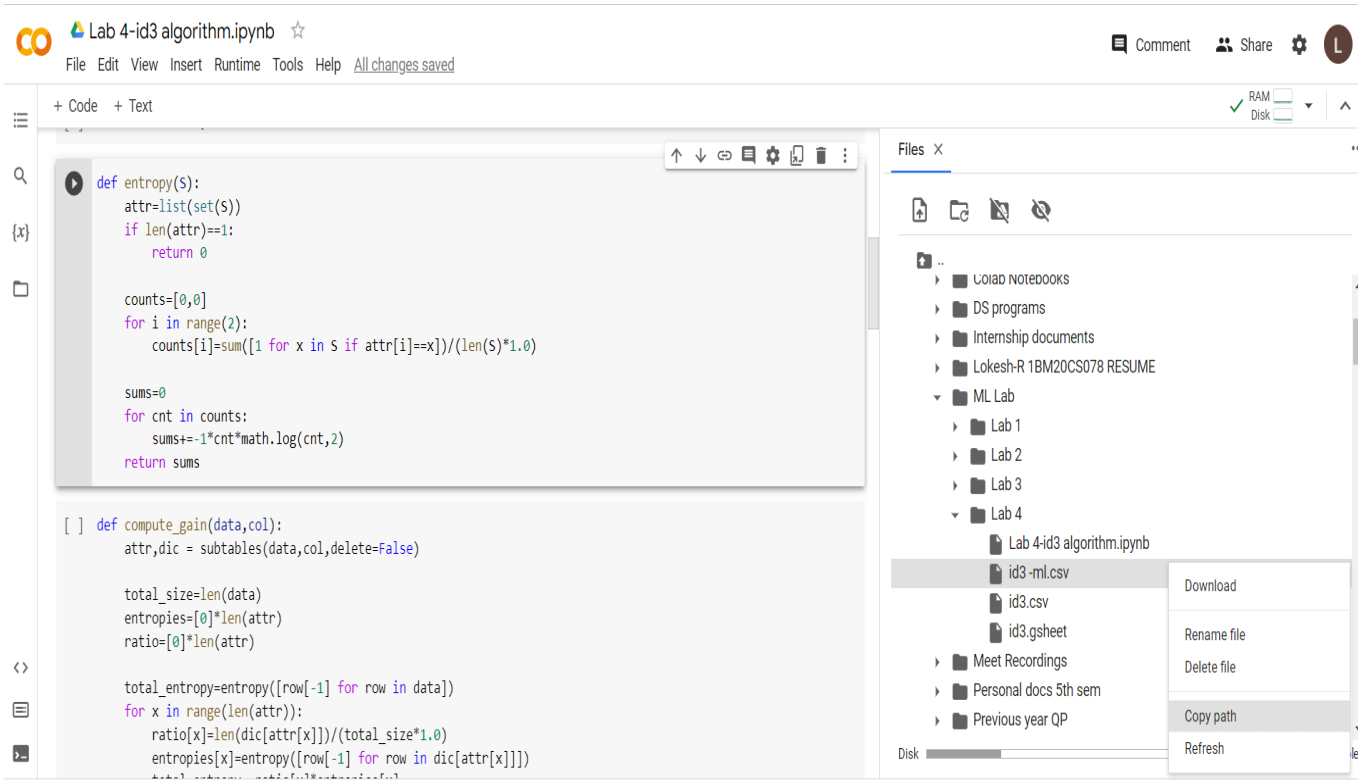
$E(S_{strong}) = 1$
$E(S_{weak}) = 0.8113$
$G(S, wind) = 0.0478$

G(outlook) is highest so it is chosen as the root node.

Uploading CSV :






Lab 4-id3 algorithm.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
def entropy(S):
attr=list(set(S))
if len(attr)==1:
return 0
counts=[0,0]
for i in range(2):
counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)
sums=0
for cnt in counts:
sums+=-1*cnt*math.log(cnt,2)
return sums
[ ] def compute_gain(data,col):
attr,dic = subtables(data,col,delete=False)
total_size=len(data)
entropies=[0]*len(attr)
ratio=[0]*len(attr)
total_entropy=entropy([row[-1] for row in data])
for x in range(len(attr)):
ratio[x]=len(dic[attr[x]])/(total_size*1.0)
entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
Files
Colab Notebooks
DS programs
Internship documents
Lokesh-R 1BM20CS078 RESUME
ML Lab
Lab 1
Lab 2
Lab 3
Lab 4
Lab 4-id3 algorithm.ipynb
id3 -ml.csv
id3.csv
id3.gsheet
Meet Recordings
Personal docs 5th sem
Previous year QP
Disk
Download
Rename file
Delete file
Copy path
Refresh

Program execution -

In [2]:
```
import math
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"))
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset,headers
```

In [4]:
```
class Node:
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=""
```

In [6]:
```
def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))

    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
        for y in range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1

    for x in range(len(attr)):
        dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
        pos=0
        for y in range(r):
            if data[y][col]==attr[x]:
                if delete:
                    del data[y][col]
                dic[attr[x]][pos]=data[y]
                pos+=1
    return attr,dic
```

In [8]:
```
def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0

    counts=[0,0]
    for i in range(2):
        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

    sums=0
    for cnt in counts:
        sums+=-1*cnt*math.log(cnt,2)
    return sums
```

In [10]:

```
def compute_gain(data,col):
    attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])
    for x in range(len(attr)):
        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
        total_entropy-=ratio[x]*entropies[x]
    return total_entropy

def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
        node=Node("")
        node.answer=lastcol[0]
        return node

    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
        gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]


    attr,dic=subtables(data,split,delete=True)

    for x in range(len(attr)):
        child=build_tree(dic[attr[x]],fea)
        node.children.append((attr[x],child))
    return node
```

In [12]:

```
def print_tree(node,level):
    if node.answer!="":
        print("  "*level,node.answer)
        return

    print("  "*level,node.attribute)
    for value,n in node.children:
        print("  "*(level+1),value)
        print_tree(n,level+2)


def classify(node,x_test,features):
    if node.answer!="":
        print(node.answer)
        return
    pos=features.index(node.attribute)
    for value, n in node.children:
        if x_test[pos]==value:
            classify(n,x_test,features)

'''Main program'''
dataset,features=load_csv("/content/drive/MyDrive/ML Lab/Lab 4/id3_ml.csv")
node1=build_tree(dataset,features)

print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
#testdata,features=load_csv("/content/drive/MyDrive/ML Lab/Lab 4/id3_ml.csv")

#for xtest in testdata:
 #   print("The test instance:",xtest)
  #  print("The label for test instance:")
   # classify(node1,xtest,features)
```

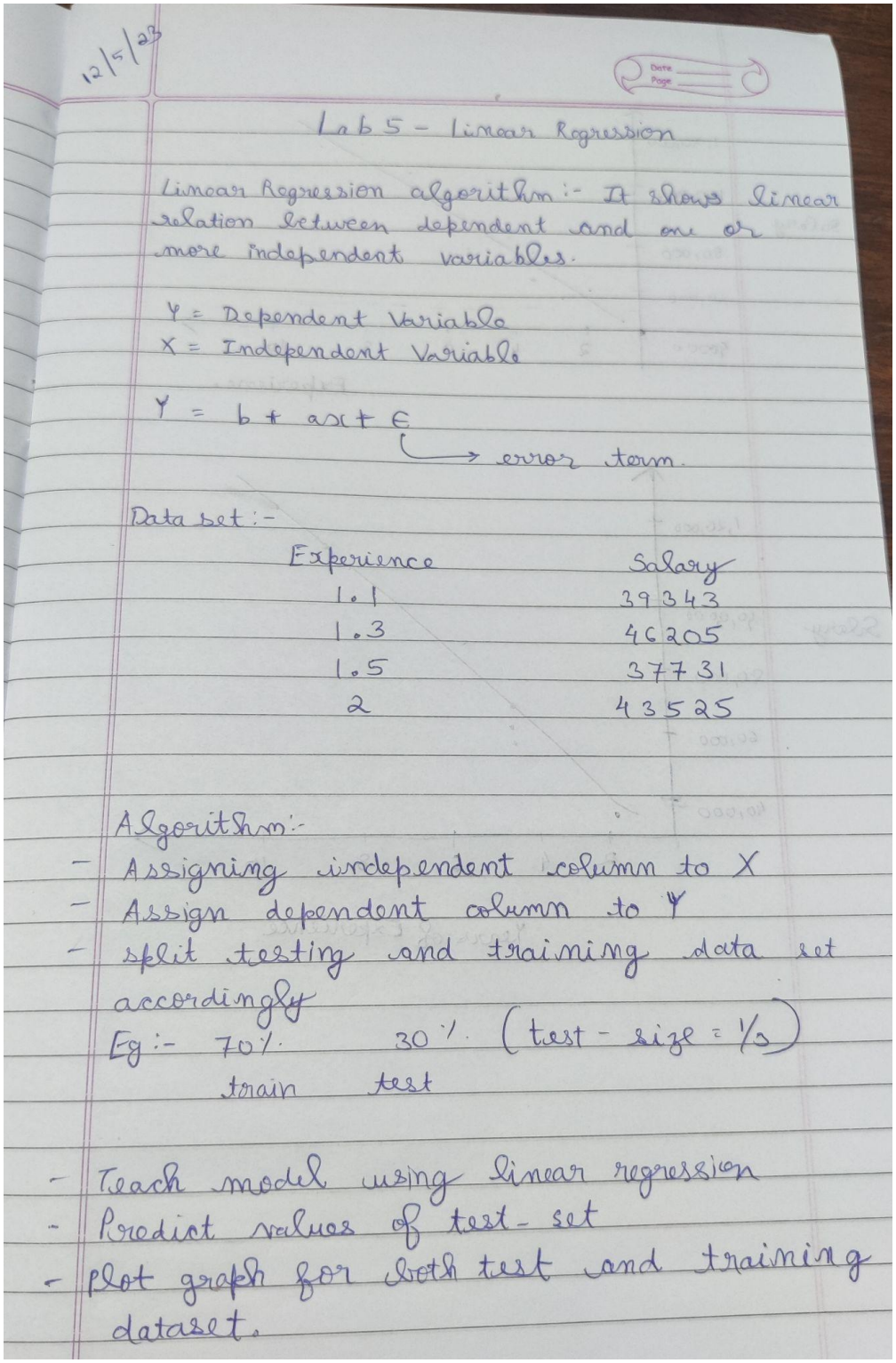
```
The decision tree for the dataset using ID3 algorithm is
 Outlook
   rain
     Wind
       strong
         no
       weak
         yes
   sunny
     Humidity
       high
         no
       normal
         yes
   overcast
     yes
```

Date:12-05-2023

Program-5: Linear regression

	A	B
1	YearsExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2	43525
6	2.2	39891
7	2.9	56642
8	3	60150
9	3.2	54445
10	3.2	64445
11	3.7	57189
12	3.9	63218
13	4	55794
14	4	56957
15	4.1	57081
16	4.5	61111
17	4.9	67938
18	5.1	66029
19	5.3	83088
20	5.9	81363
21	6	93940
22	6.8	91738
23	7.1	98273
24	7.9	101302
25	8.2	113812
26	8.7	109431
27	9	105582
28	9.5	116969
29	9.6	112635
30	10.3	122391
31	10.5	121872

Observation :



12/5/23

Lab 5 - Linear Regression

Linear Regression algorithm :- It shows linear relation between dependent and one or more independent variables.

Y = Dependent Variable
X = Independent Variable

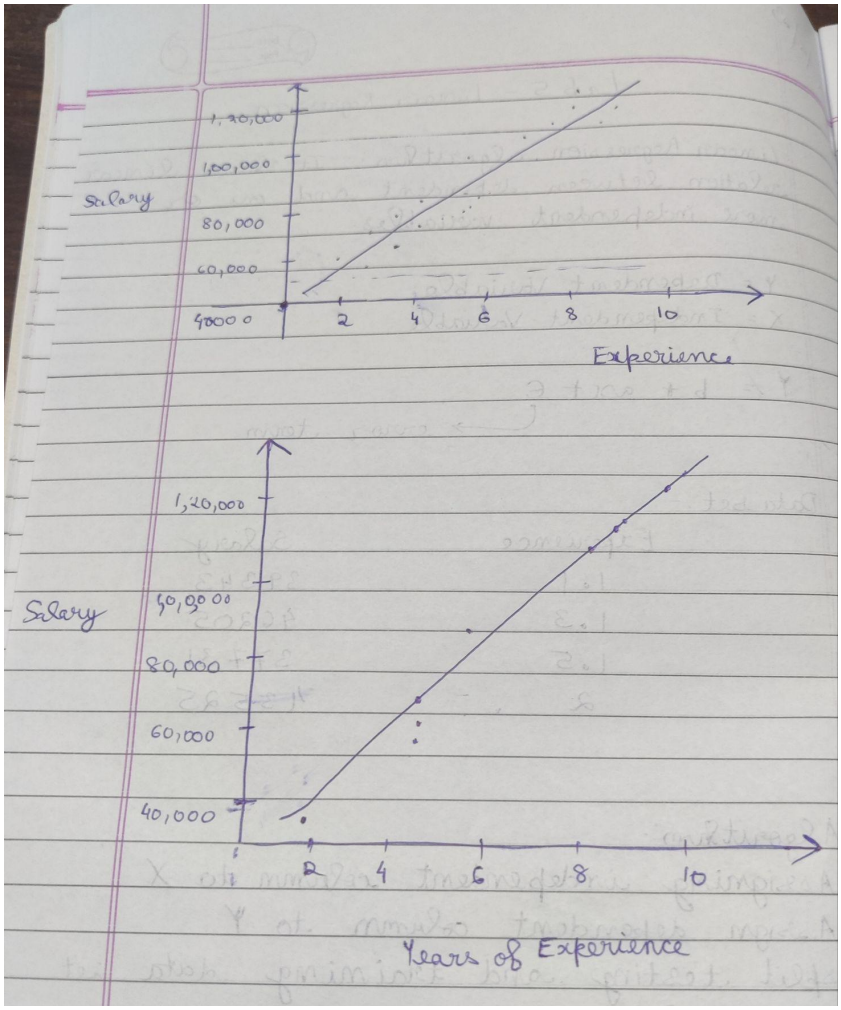
$Y = b + ax + \epsilon$ → error term.

Data set :-

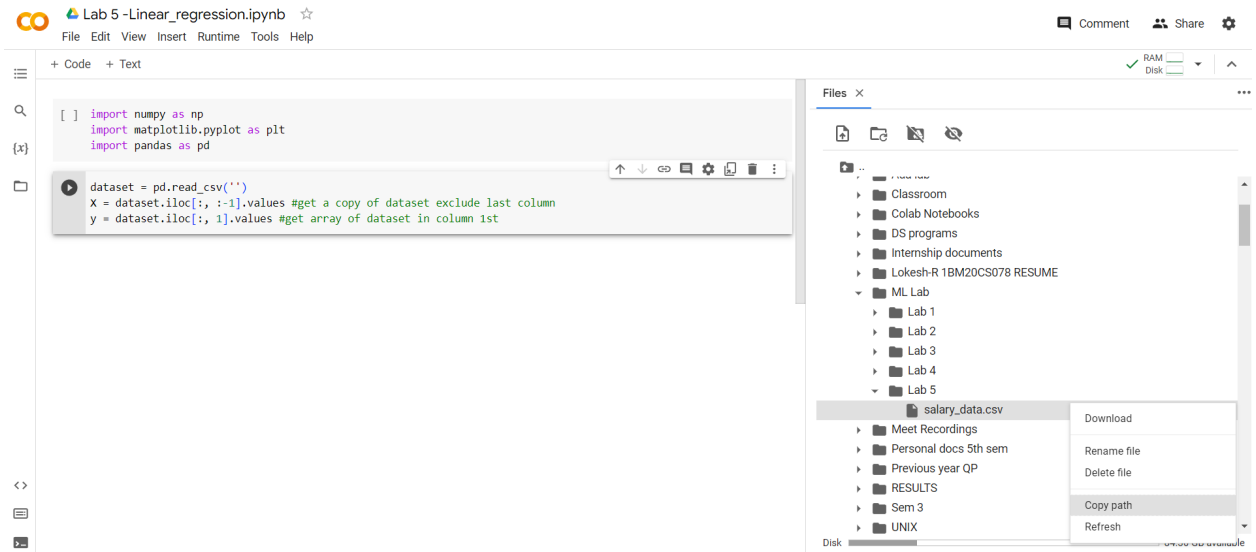
Experience	Salary
1.1	39343
1.3	46205
1.5	37731
2	43525

Algorithm :-
- Assigning independent column to X
- Assign dependent column to Y
- split testing and training data set accordingly
  Eg :- 70% 30% (test - size = 1/3)
  train test

- Teach model using linear regression
- Predict values of test - set
- plot graph for both test and training dataset.



Uploading CSV :



Program execution-

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
dataset = pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 5/salary_data.csv')
X = dataset.iloc[:, :-1].values #get a copy of dataset exclude last column
y = dataset.iloc[:, 1].values #get array of dataset in column 1st
```

In [4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)



# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```


Salary VS Experience (Training set)
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Year of Experience


Salary VS Experience (Test set)
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Year of Experience

Date: 19.05.2023

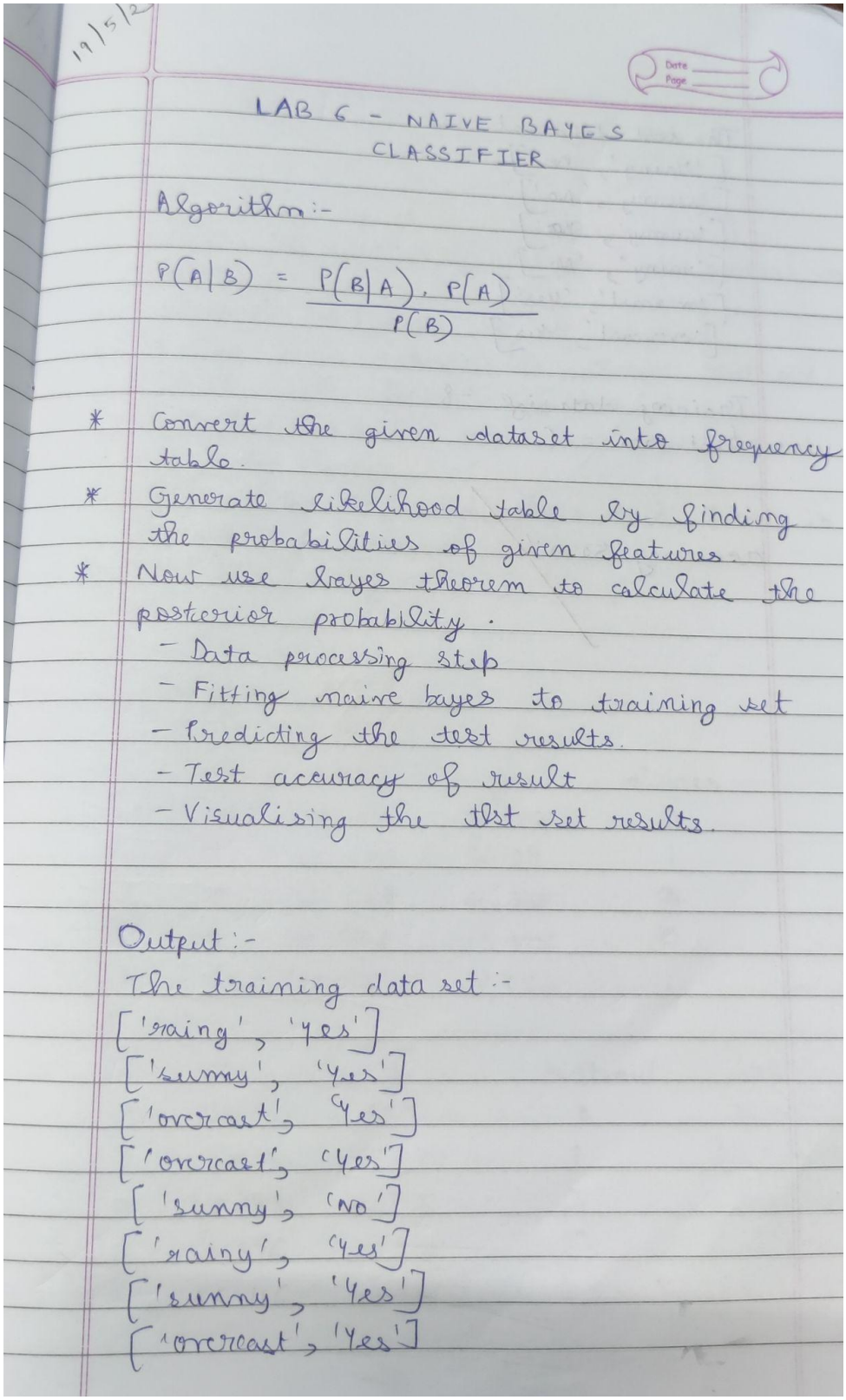
Program 6 – Write a program to construct a Bayesian network considering training data. Use this model to make predictions.

Data set: Climate Dataset

nbc.csv

	A	B
1	outlook	play
2	rainy	Yes
3	sunny	Yes
4	overcast	Yes
5	overcast	Yes
6	sunny	No
7	rainy	Yes
8	sunny	Yes
9	overcast	Yes
10	rainy	No
11	sunny	No
12	sunny	Yes
13	rainy	No
14	overcast	Yes
15	overcast	Yes

Observation :



19/5/2

LAB 6 - NAIVE BAYES CLASSIFIER

Algorithm :-

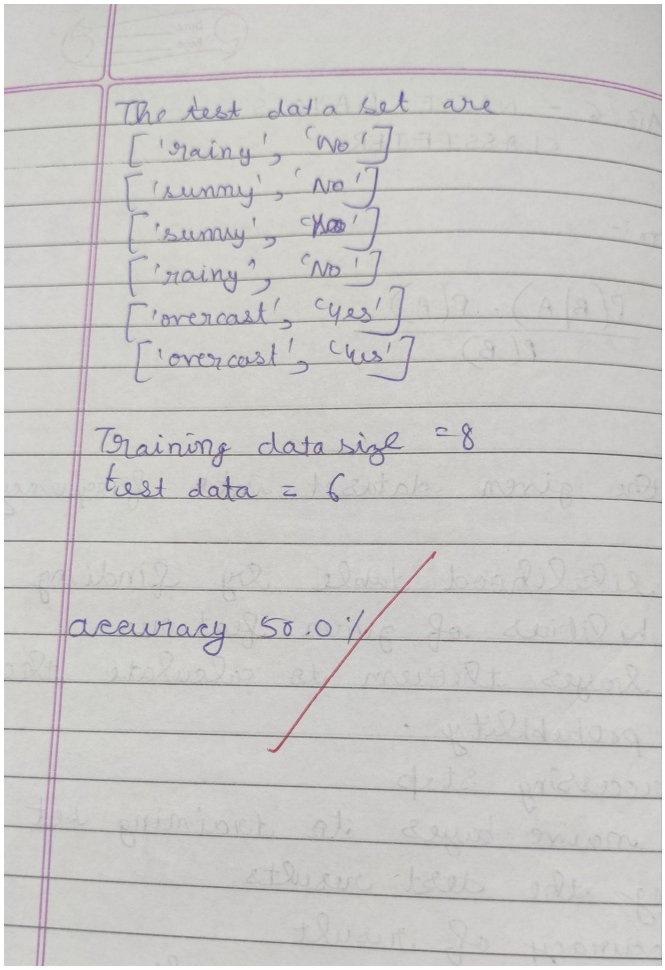
$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

* Convert the given dataset into frequency table.
* Generate likelihood table by finding the probabilities of given features.
* Now use bayes theorem to calculate the posterior probability.
  - Data processing step
  - Fitting naive bayes to training set
  - Predicting the test results.
  - Test accuracy of result
  - Visualising the test set results.

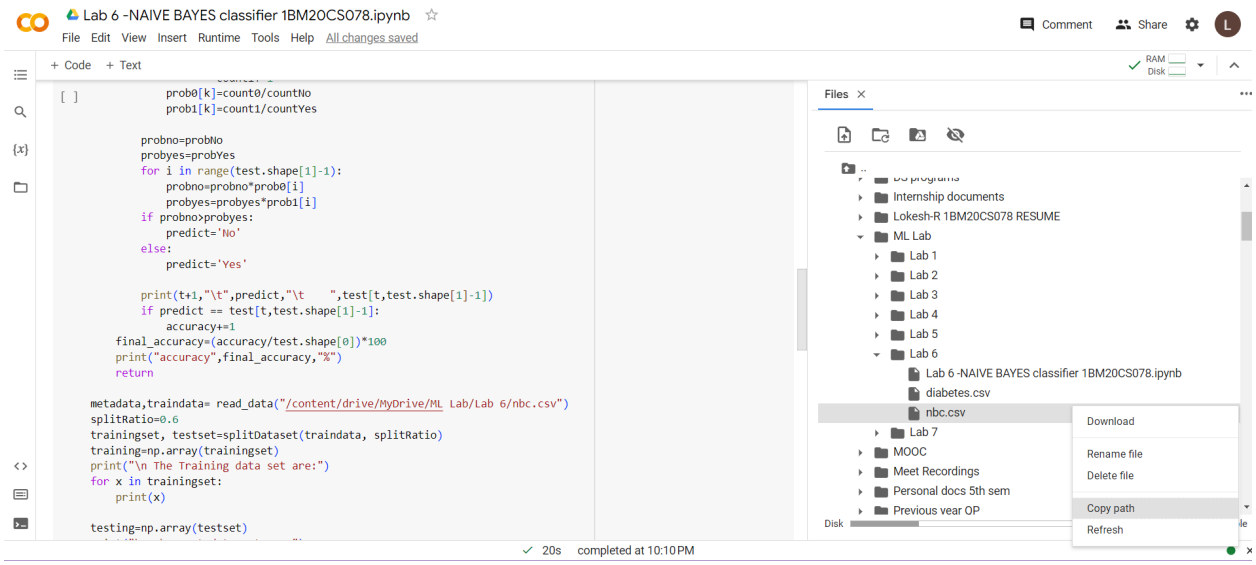
Output :-

The training data set :-

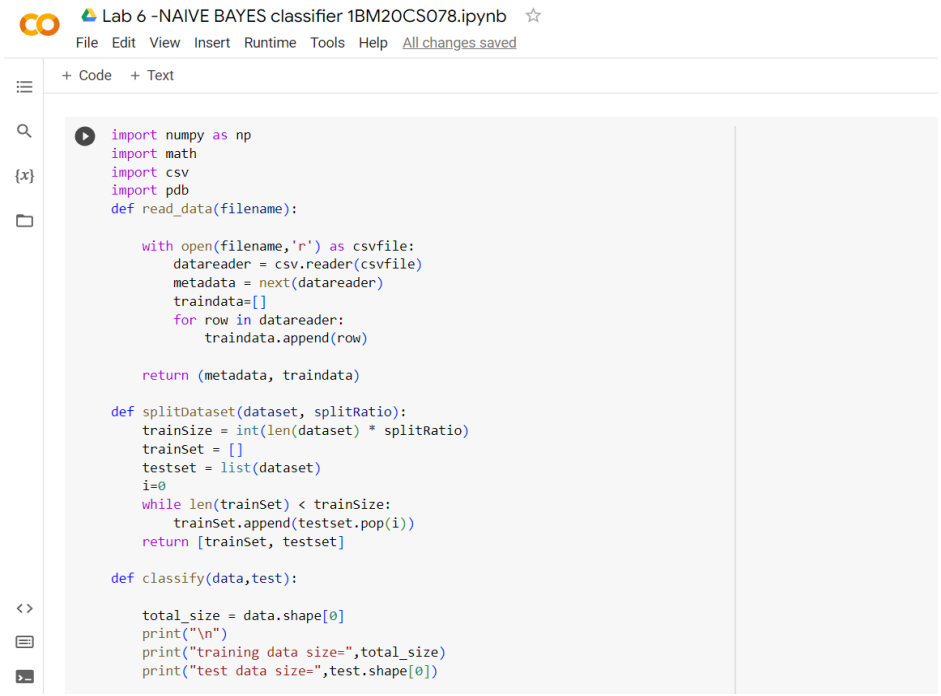
['rainy', 'yes']
['sunny', 'yes']
['overcast', 'yes']
['overcast', 'yes']
['sunny', 'NO']
['rainy', 'yes']
['sunny', 'yes']
['overcast', 'yes']


The test data set are
['rainy', 'No']
['sunny', 'No']
['sunny', 'No']
['rainy', 'No']
['overcast', 'Yes']
['overcast', 'Yes']
Training data size = 8
test data = 6
accuracy 50.0 %

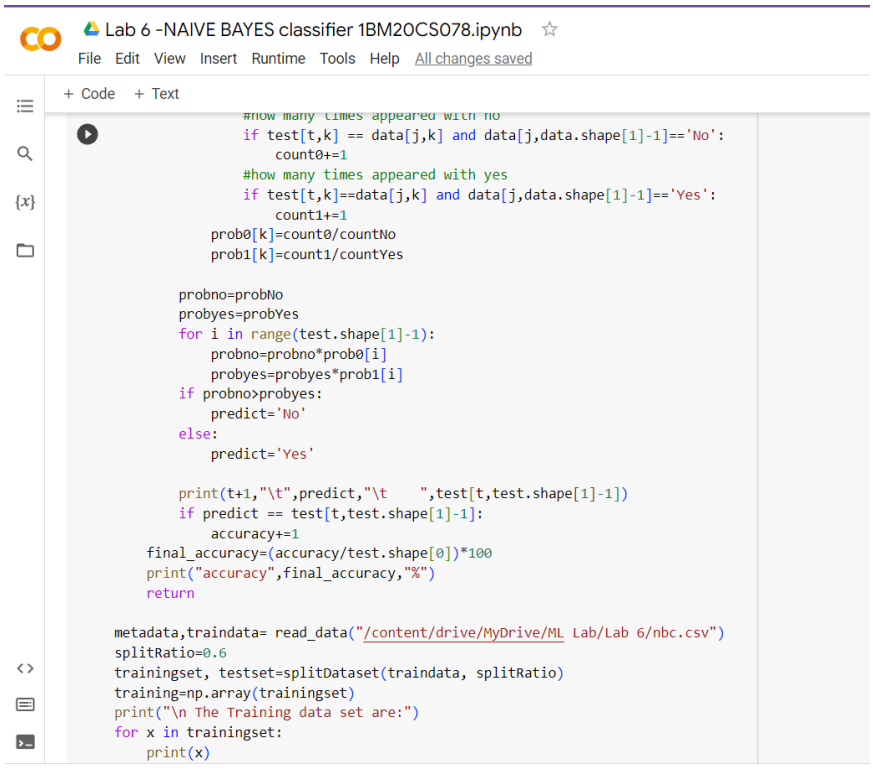
Uploading csv file:

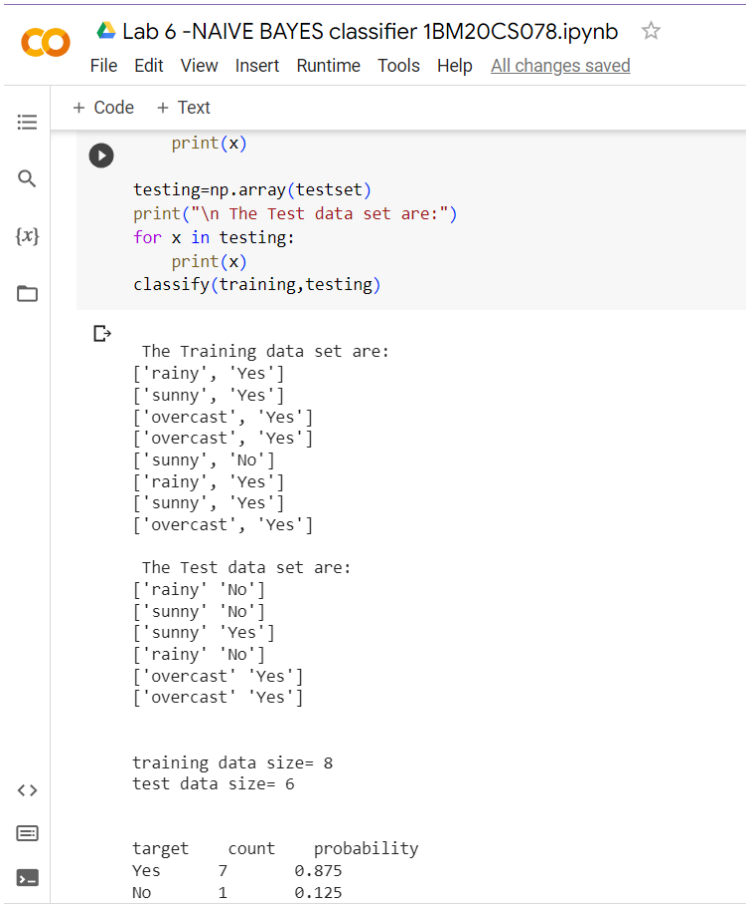


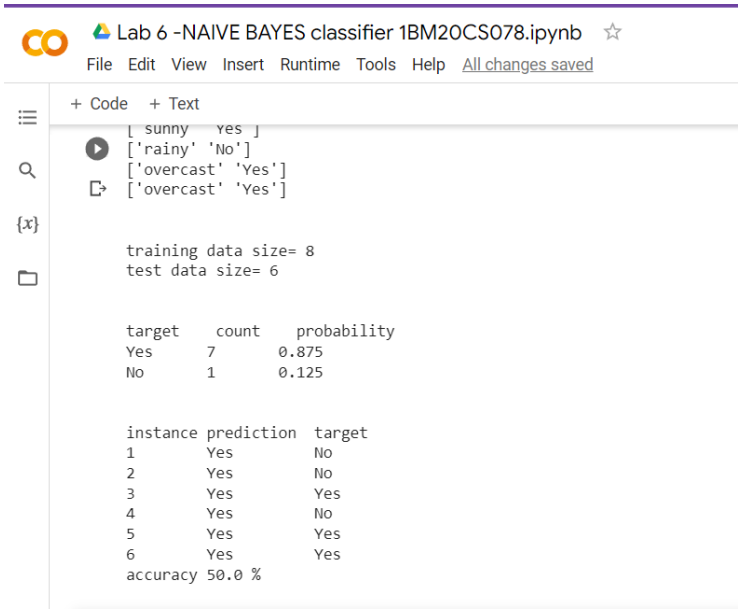
Program execution-




Lab 6 -NAIVE BAYES classifier 1BM20CS078.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
countYes = 0
countNo = 0
probYes = 0
probNo = 0
print("\n")
print("target count probability")
for x in range(data.shape[0]):
if data[x,data.shape[1]-1] == 'Yes':
countYes +=1
if data[x,data.shape[1]-1] == 'No':
countNo +=1
probYes=countYes/total_size
probNo= countNo / total_size
print('Yes',"\t",countYes,"\t",probYes)
print('No',"\t",countNo,"\t",probNo)
prob0 =np.zeros((test.shape[1]-1))
prob1 =np.zeros((test.shape[1]-1))
accuracy=0
print("\n")
print("instance prediction target")
for t in range(test.shape[0]):
for k in range (test.shape[1]-1):
count1=count0=0
for j in range (data.shape[0]):
#how many times appeared with no


Lab 6 -NAIVE BAYES classifier 1BM20CS078.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
#how many times appeared with no
if test[t,k] == data[j,k] and data[j,data.shape[1]-1]=='No':
count0+=1
#how many times appeared with yes
if test[t,k]==data[j,k] and data[j,data.shape[1]-1]=='Yes':
count1+=1
prob0[k]=count0/countNo
prob1[k]=count1/countYes
probno=probNo
probyes=probYes
for i in range(test.shape[1]-1):
probno=probno*prob0[i]
probyes=probyes*prob1[i]
if probno>probyes:
predict='No'
else:
predict='Yes'
print(t+1,"\t",predict,"\t ",test[t,test.shape[1]-1])
if predict == test[t,test.shape[1]-1]:
accuracy+=1
final_accuracy=(accuracy/test.shape[0])*100
print("accuracy",final_accuracy,"%")
return
metadata,traindata= read_data("/content/drive/MyDrive/ML Lab/Lab 6/nbc.csv")
splitRatio=0.6
trainingset, testset=splitDataset(traindata, splitRatio)
training=np.array(trainingset)
print("\n The Training data set are:")
for x in trainingset:
print(x)

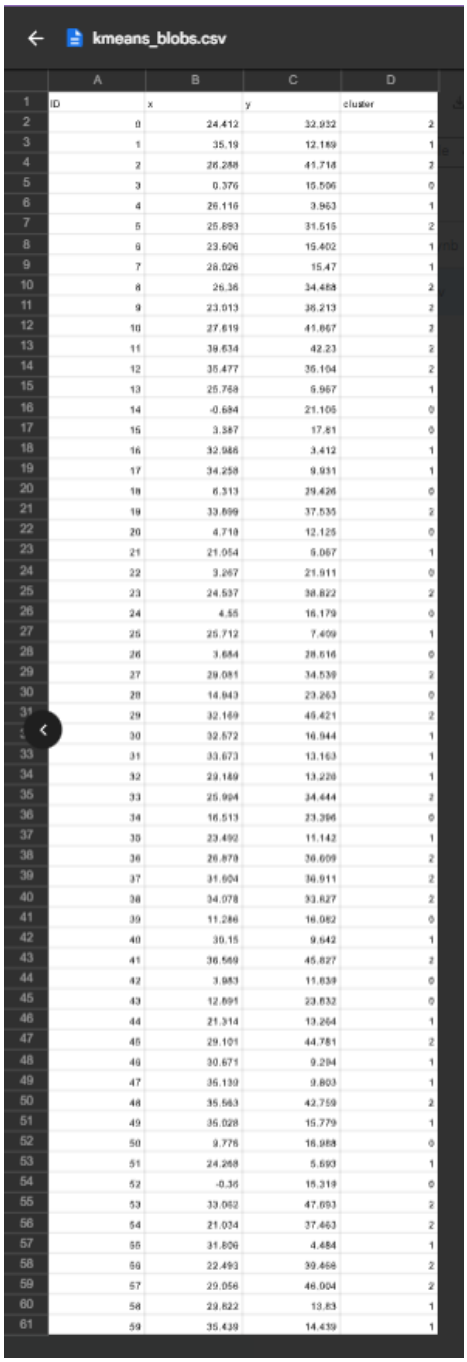

Lab 6 -NAIVE BAYES classifier 1BM20CS078.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
print(x)
testing=np.array(testset)
print("\n The Test data set are:")
for x in testing:
print(x)
classify(training,testing)
The Training data set are:
['rainy', 'Yes']
['sunny', 'Yes']
['overcast', 'Yes']
['overcast', 'Yes']
['sunny', 'No']
['rainy', 'Yes']
['sunny', 'Yes']
['overcast', 'Yes']
The Test data set are:
['rainy' 'No']
['sunny' 'No']
['sunny' 'Yes']
['rainy' 'No']
['overcast' 'Yes']
['overcast' 'Yes']
training data size= 8
test data size= 6
target count probability
Yes 7 0.875
No 1 0.125


Lab 6 -NAIVE BAYES classifier 1BM20CS078.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
['rainy' 'No']
['overcast' 'Yes']
['overcast' 'Yes']
training data size= 8
test data size= 6
target count probability
Yes 7 0.875
No 1 0.125
instance prediction target
1 Yes No
2 Yes No
3 Yes Yes
4 Yes No
5 Yes Yes
6 Yes Yes
accuracy 50.0 %

Date: 19.05.2023

Program 7 – Apply k-Means algorithm to cluster a set of data stored in a .CSV file.

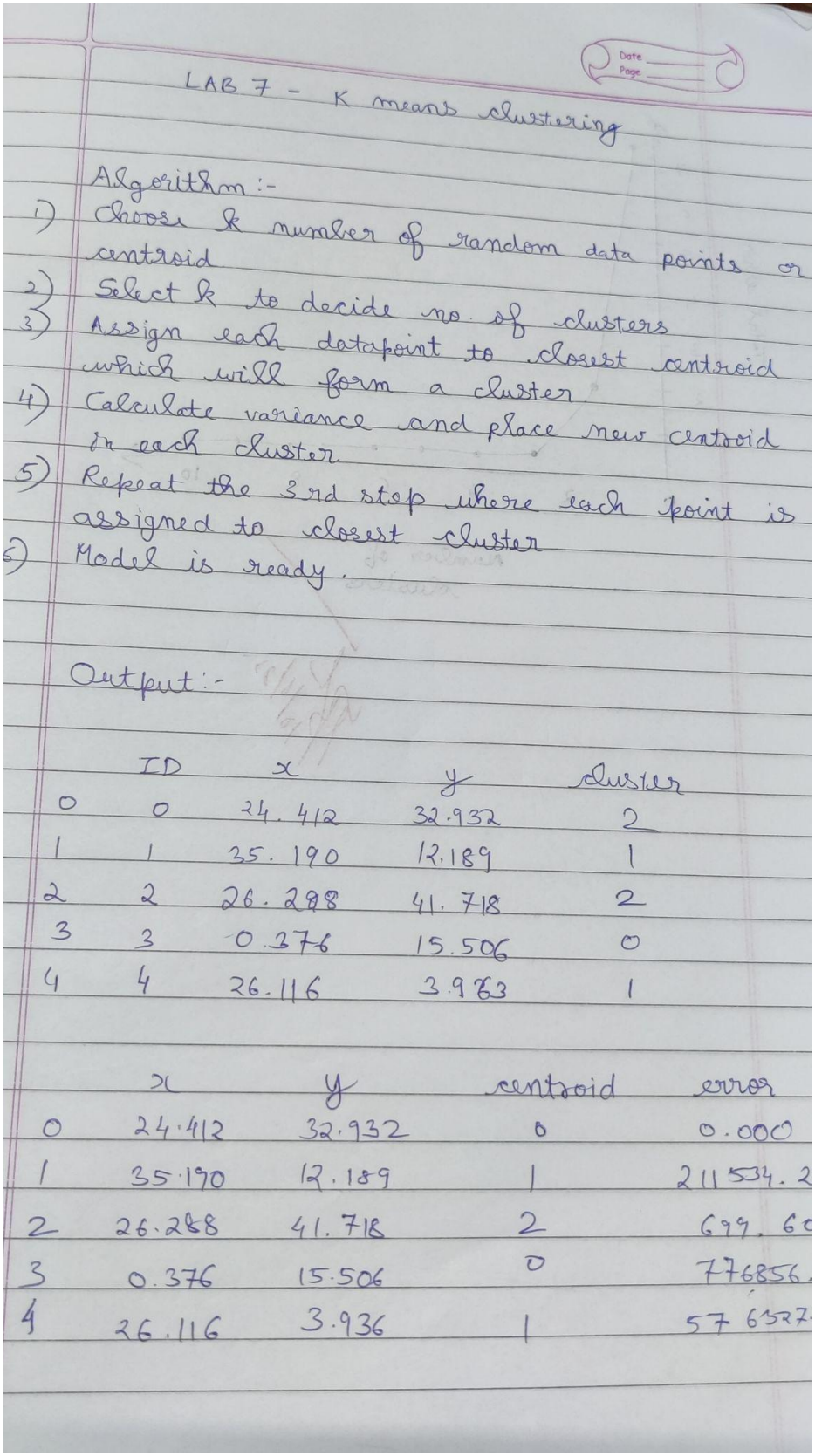
Data set:



kmeans_blobs.csv

	A	B	C	D
1	ID	x	y	cluster
2	0	24.412	32.932	2
3	1	35.19	12.189	1
4	2	26.288	41.718	2
5	3	0.376	15.506	0
6	4	26.116	3.963	1
7	5	25.893	31.515	2
8	6	23.606	15.402	1
9	7	28.026	15.47	1
10	8	26.36	34.488	2
11	9	23.013	36.213	2
12	10	27.619	41.667	2
13	11	39.634	42.23	2
14	12	35.477	36.104	2
15	13	25.768	6.967	1
16	14	-0.684	21.105	0
17	15	3.387	17.61	0
18	16	32.986	3.412	1
19	17	34.258	9.931	1
20	18	6.313	29.426	0
21	19	33.899	37.535	2
22	20	4.718	12.125	0
23	21	21.054	6.067	1
24	22	3.267	21.911	0
25	23	24.537	38.822	2
26	24	4.55	16.179	0
27	25	25.712	7.409	1
28	26	3.684	28.616	0
29	27	29.081	34.539	2
30	28	14.943	23.263	0
31	29	32.169	46.421	2
32	30	32.572	16.944	1
33	31	33.673	13.163	1
34	32	29.189	13.228	1
35	33	25.994	34.444	2
36	34	16.513	23.396	0
37	35	23.492	11.142	1
38	36	26.879	36.609	2
39	37	31.604	36.911	2
40	38	34.078	33.627	2
41	39	11.286	16.082	0
42	40	30.15	9.642	1
43	41	36.569	45.827	2
44	42	3.983	11.639	0
45	43	12.891	23.832	0
46	44	21.314	13.264	1
47	45	29.101	44.781	2
48	46	30.671	9.294	1
49	47	36.139	9.803	1
50	48	35.563	42.759	2
51	49	35.028	15.779	1
52	50	9.776	16.988	0
53	51	24.268	5.693	1
54	52	-0.36	15.319	0
55	53	33.062	47.093	2
56	54	21.034	37.463	2
57	55	31.806	4.484	1
58	56	22.493	39.468	2
59	57	29.056	46.004	2
60	58	29.822	13.83	1
61	59	35.439	14.439	1

Observation :



LAB 7 - K means clustering

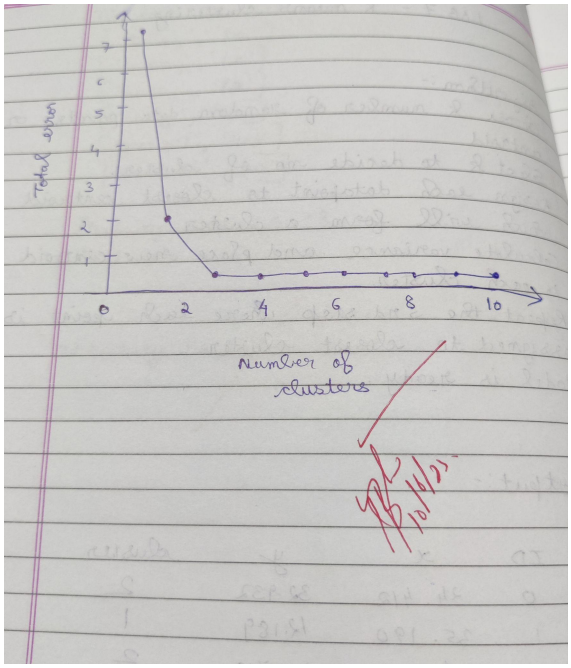
Algorithm :-

1) Choose k number of random data points or centroid
2) Select k to decide no. of clusters
3) Assign each datapoint to closest centroid which will form a cluster
4) Calculate variance and place new centroid in each cluster
5) Repeat the 3rd step where each point is assigned to closest cluster
6) Model is ready.

Output :-

	ID	x	y	cluster
0	0	24.412	32.932	2
1	1	35.190	12.189	1
2	2	26.288	41.718	2
3	3	0.376	15.506	0
4	4	26.116	3.983	1

	x	y	centroid	error
0	24.412	32.932	0	0.000
1	35.190	12.189	1	211534.2
2	26.288	41.718	2	699.60
3	0.376	15.506	0	776856.
4	26.116	3.936	1	57 6327



Program execution:

Open in Colab

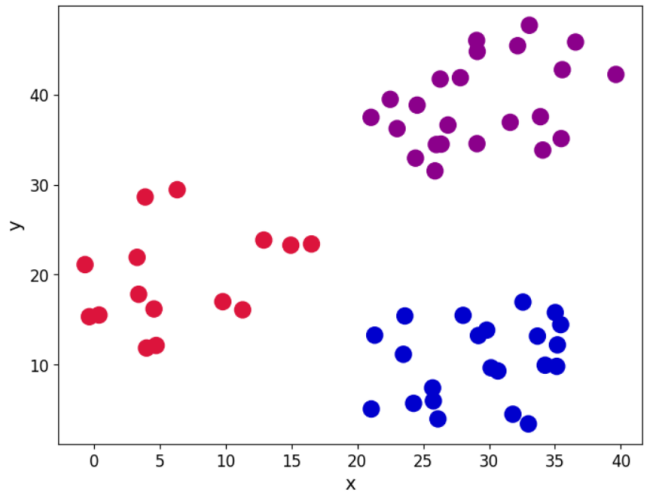
In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
%matplotlib inline

blobs = pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 6/kmeans_blobs.csv')
colnames = list(blobs.columns[1:-1])
blobs.head()
```

Out[ ]:

	ID	x	y	cluster
0	0	24.412	32.932	2
1	1	35.190	12.189	1
2	2	26.288	41.718	2
3	3	0.376	15.506	0
4	4	26.116	3.963	1

In [ ]:
```
customcmap = ListedColormap(["crimson", "mediumblue", "darkmagenta"])

fig, ax = plt.subplots(figsize=(8, 6))
plt.scatter(x=blobs['x'], y=blobs['y'], s=150,
            c=blobs['cluster'].astype('category'),
            cmap = customcmap)
ax.set_xlabel(r'x', fontsize=14)
ax.set_ylabel(r'y', fontsize=14)
```

```
plt.yticks(fontsize=12)
plt.show()
```



In [ ]:
```
def initiate_centroids(k, dset):
    '''
    Select k data points as centroids
    k: number of centroids
    dset: pandas dataframe
    '''
    centroids = dset.sample(k)
    return centroids

np.random.seed(42)
k=3
df = blobs[['x','y']]
centroids = initiate_centroids(k, df)
centroids
```

Out[ ]:

	x	y
0	24.412	32.932
5	25.893	31.515
36	26.878	36.609

In [ ]:
```
def rsserr(a,b):
    '''
    Calculate the root of sum of squared errors.
    a and b are numpy arrays
    '''
    return np.square(np.sum((a-b)**2))
```

In [ ]:
```
for i, centroid in enumerate(range(centroids.shape[0])):
    err = rsserr(centroids.iloc[centroid,:], df.iloc[36,:])
    print('Error for centroid {0}: {1:.2f}'.format(i, err))
```

```
Error for centroid 0: 384.22
Error for centroid 1: 724.64
Error for centroid 2: 0.00
```

In [ ]:
```
def centroid_assignation(dset, centroids):
    '''
    Given a dataframe `dset` and a set of `centroids`, we assign each
    data point in `dset` to a centroid.
    - dset - pandas dataframe with observations
    - centroids - pa das dataframe with centroids
    '''
    k = centroids.shape[0]
    n = dset.shape[0]
    assignation = []
    assign_errors = []
    for obs in range(n):
        # Estimate error
        all_errors = np.array([])
```

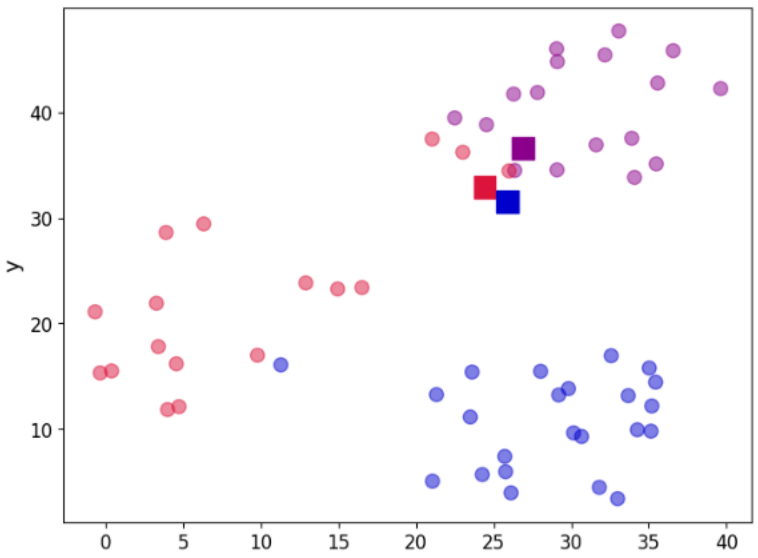
```
        for centroid in range(k):
            err = rsserr(centroids.iloc[centroid, :], dset.iloc[obs,:])
            all_errors = np.append(all_errors, err)

        # Get the nearest centroid and the error
        nearest_centroid =  np.where(all_errors==np.amin(all_errors))[0].tolist()[0]
        nearest_centroid_error = np.amin(all_errors)

        # Add values to corresponding lists
        assignation.append(nearest_centroid)
        assign_errors.append(nearest_centroid_error)
    return assignation, assign_errors
```

In [8]:
```
df['centroid'], df['error'] = centroid_assignation(df, centroids)
df.head()
```

Out[8]:

	x	y	centroid	error
0	24.412	32.932	0	0.000000
1	35.190	12.189	1	211534.211314
2	26.288	41.718	2	699.601495
3	0.376	15.506	0	776856.744109
4	26.116	3.963	1	576327.599678

In [9]:
```
fig, ax = plt.subplots(figsize=(8, 6))
plt.scatter(df.iloc[:,0], df.iloc[:,1],  marker = 'o',
            c=df['centroid'].astype('category'),
            cmap = customcmap, s=80, alpha=0.5)
plt.scatter(centroids.iloc[:,0], centroids.iloc[:,1],
            marker = 's', s=200, c=[0, 1, 2],
            cmap = customcmap)
ax.set_xlabel(r'x', fontsize=14)
ax.set_ylabel(r'y', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



In [10]:
```
print("The total error is {0:.2f}".format(df['error'].sum()))
```

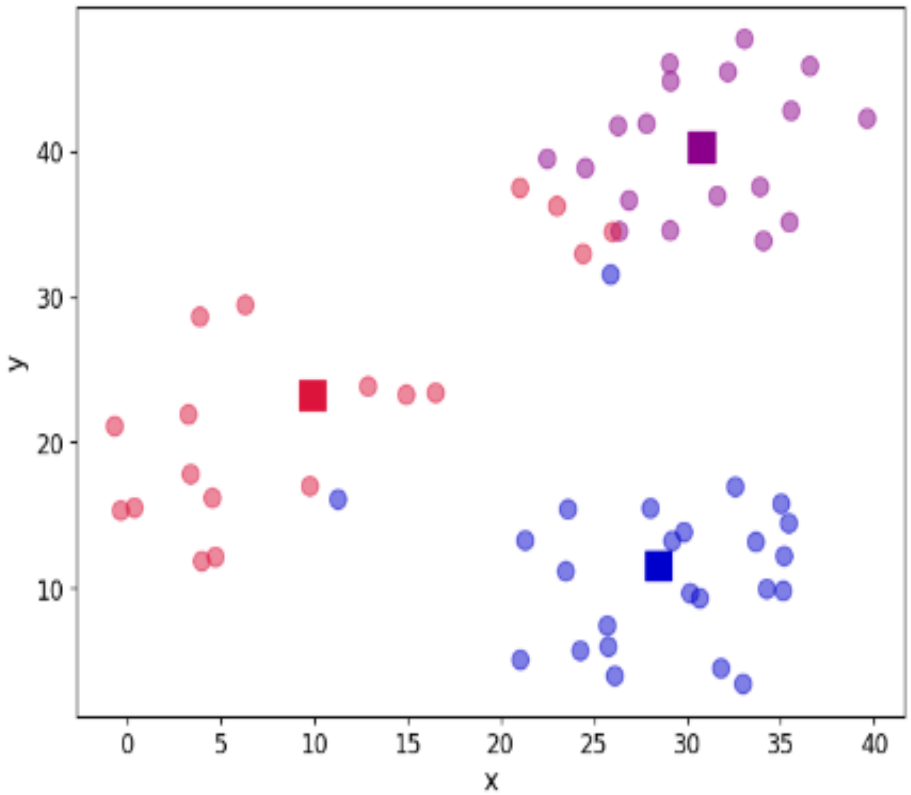
```
The total error is 11927659.01
```

In [11]:
```
centroids = df.groupby('centroid').agg('mean').loc[:, colnames].reset_index(drop = True)
centroids
```

Out[11]:

	x	y
0	9.889444	23.242611
1	28.435750	11.546250
2	30.759333	40.311167

In [12]:
```
fig, ax = plt.subplots(figsize=(8, 6))
plt.scatter(df.iloc[:,0], df.iloc[:,1],  marker = 'o',
            c=df['centroid'].astype('category'),
            cmap = customcmap, s=80, alpha=0.5)
plt.scatter(centroids.iloc[:,0], centroids.iloc[:,1],
            marker = 's', s=200,
            c=[0, 1, 2], cmap = customcmap)
ax.set_xlabel(r'x', fontsize=14)
ax.set_ylabel(r'y', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



In [13]:

```
def kmeans(dset, k=2, tol=1e-4):
    '''
    K-means implementationd for a
    `dset`:  DataFrame with observations
    `k`: number of clusters, default k=2
    `tol`: tolerance=1E-4
    '''
    # Let us work in a copy, so we don't mess the original
    working_dset = dset.copy()
    # We define some variables to hold the error, the
    # stopping signal and a counter for the iterations
    err = []
    goahead = True
    j = 0

    # Step 2: Initiate clusters by defining centroids
    centroids = initiate_centroids(k, dset)
    while(goahead):
        # Step 3 and 4 - Assign centroids and calculate error
        working_dset['centroid'], j_err = centroid_assignation(working_dset, centroids)
        err.append(sum(j_err))

        # Step 5 - Update centroid position
        centroids = working_dset.groupby('centroid').agg('mean').reset_index(drop = True)

        # Step 6 - Restart the iteration
        if j>0:
            # Is the error less than a tolerance (1E-4)
            if err[j-1]-err[j]<=tol:
                goahead = False
        j+=1

    working_dset['centroid'], j_err = centroid_assignation(working_dset, centroids)
    centroids = working_dset.groupby('centroid').agg('mean').reset_index(drop = True)
    return working_dset['centroid'], j_err, centroids
```

In [14]:

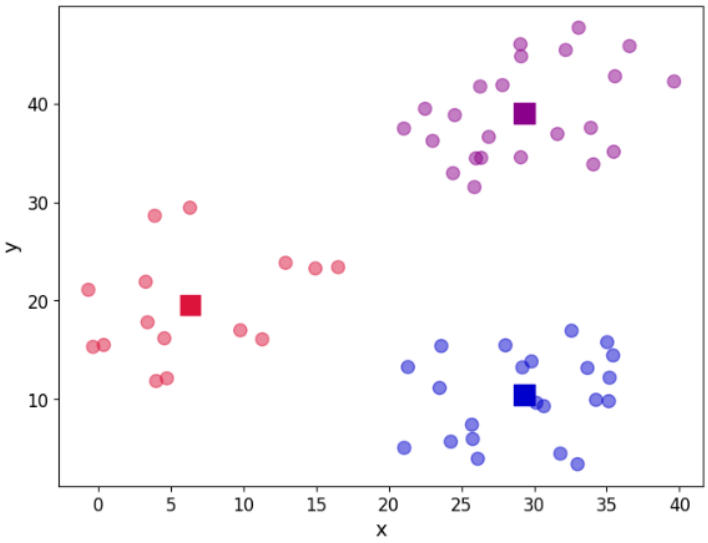
```
np.random.seed(42)
df['centroid'], df['error'], centroids =  kmeans(df[['x','y']], 3)
df.head()
```

Out[14]:

	x	y	centroid	error
0	24.412	32.932	2	3767.568743
1	35.190	12.189	1	1399.889001
2	26.288	41.718	2	262.961097
3	0.376	15.506	0	2683.086425
4	26.116	3.963	1	2723.650198

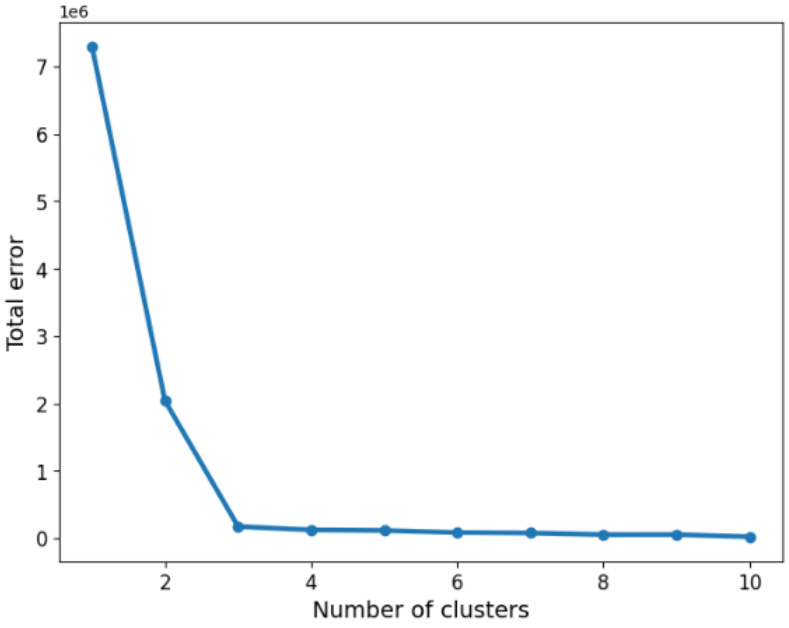
In [15]:

```
fig, ax = plt.subplots(figsize=(8, 6))
plt.scatter(df.iloc[:,0], df.iloc[:,1],  marker = 'o',
            c=df['centroid'].astype('category'),
            cmap = customcmap, s=80, alpha=0.5)
plt.scatter(centroids.iloc[:,0], centroids.iloc[:,1],
            marker = 's', s=200, c=[0, 1, 2],
            cmap = customcmap)
ax.set_xlabel(r'x', fontsize=14)
ax.set_ylabel(r'y', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



In [16]:

```
err_total = []
n = 10

df_elbow = blobs[['x','y']]

for i in range(n):
    _, my_errs, _ = kmeans(df_elbow, i+1)
    err_total.append(sum(my_errs))
fig, ax = plt.subplots(figsize=(8, 6))
plt.plot(range(1,n+1), err_total, linewidth=3, marker='o')
ax.set_xlabel(r'Number of clusters', fontsize=14)
ax.set_ylabel(r'Total error', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

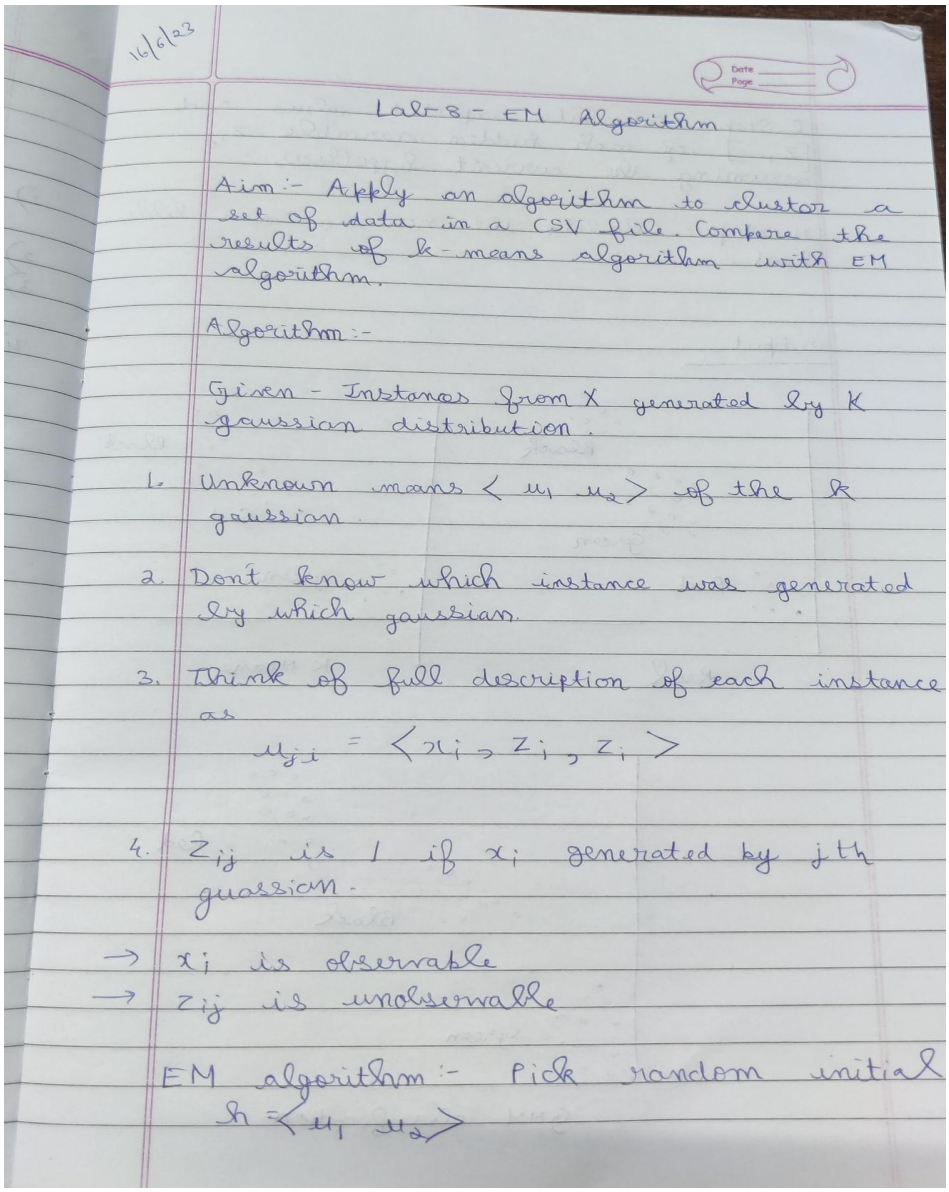


Date: 10.06.2023

Program 8 – Apply EM algorithm to cluster a set of data stored in a .CSV file. Compare the results of k-Means algorithm and EM algorithm.

Data set: Iris Dataset

Observation:



16/6/23

Lab 8 - EM Algorithm

Aim :- Apply an algorithm to cluster a set of data in a CSV file. Compare the results of k-means algorithm with EM algorithm.

Algorithm :-

Given - Instances from X generated by K gaussian distribution.

1. Unknown means $\langle u_1 \; u_2 \rangle$ of the k gaussian
2. Don't know which instance was generated by which gaussian.
3. Think of full description of each instance as

$$y_{ji} = \langle x_i, z_i, z_i \rangle$$

4. z_{ij} is 1 if x_i generated by jth gaussian.

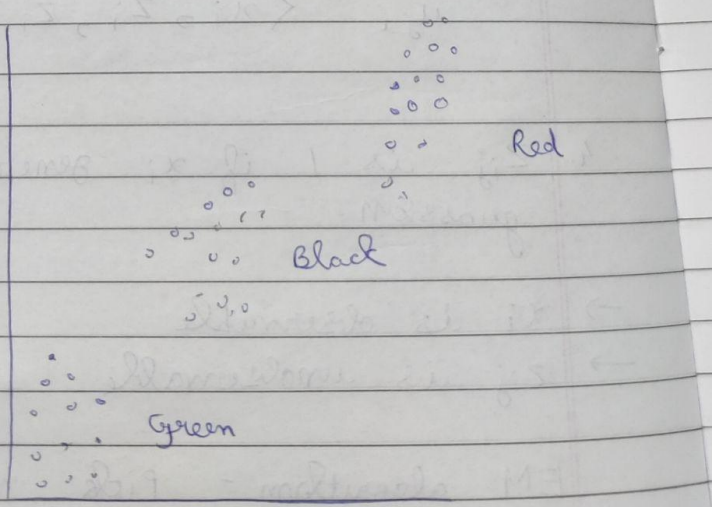
→ x_i is observable
→ z_{ij} is unobservable

EM algorithm :- Pick random initial $h = \langle u_1 \; u_2 \rangle$

E Step :- Calculate expected values and $[z_{ij}]$ of each hidden variable z_{ij} assuming the current hypothesis.

M step :- Calculate the new maximum likelihood.

Output :-



Program execution :

In [1]:

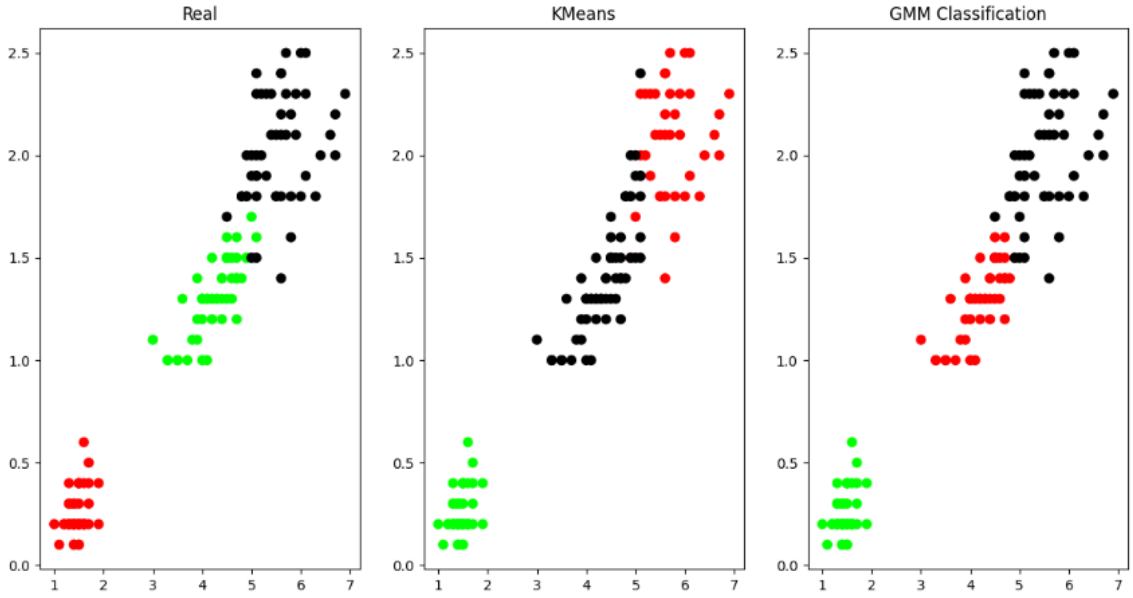
```
import numpy as np
xs = np.array([(5,5), (9,1), (8,2), (4,6), (7,3)])
thetas = np.array([[0.6, 0.4], [0.5, 0.5]])

tol = 0.01
max_iter = 100

ll_old = 0
for i in range(max_iter):
    ws_A = []
    ws_B = []

    vs_A = []
    vs_B = []

    ll_new = 0

    # E-step: calculate probability distributions over possible completions
    for x in xs:

        # multinomial (binomial) log likelihood
        ll_A = np.sum([x*np.log(thetas[0])])
        ll_B = np.sum([x*np.log(thetas[1])])

        # [EQN 1]
        denom = np.exp(ll_A) + np.exp(ll_B)
        w_A = np.exp(ll_A)/denom
        w_B = np.exp(ll_B)/denom

        ws_A.append(w_A)
        ws_B.append(w_B)
```

```
        # used for calculating theta
        vs_A.append(np.dot(w_A, x))
        vs_B.append(np.dot(w_B, x))

        # update complete log likelihood
        ll_new += w_A * ll_A + w_B * ll_B

    # M-step: update values for parameters given current distribution
    # [EQN 2]
    thetas[0] = np.sum(vs_A, 0)/np.sum(vs_A)
    thetas[1] = np.sum(vs_B, 0)/np.sum(vs_B)
    # print distribution of z for each x and current parameter estimate

    print ("Iteration: %d" % (i+1))
    print ("theta_A = %.2f, theta_B = %.2f, ll = %.2f" % (thetas[0,0], thetas[1,0], ll_new))

    if np.abs(ll_new - ll_old) < tol:
        break
    ll_old = ll_new

from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=load_iris()
# print(dataset)
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
# print(X)
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
```

```
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')

# K-PLOT
plt.subplot(1,3,2)
model=KMeans(n_clusters=3)
model.fit(X)
predY=np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)
plt.title('KMeans')

# GMM PLOT
scaler=preprocessing.StandardScaler()
scaler.fit(X)
xsa=scaler.transform(X)
xs=pd.DataFrame(xsa,columns=X.columns)
gmm=GaussianMixture(n_components=3)
gmm.fit(xs)
y_cluster_gmm=gmm.predict(xs)
plt.subplot(1,3,3)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)
plt.title('GMM Classification')
```

```
Iteration: 1
theta_A = 0.71, theta_B = 0.58, ll = -32.69
Iteration: 2
theta_A = 0.75, theta_B = 0.57, ll = -31.26
Iteration: 3
theta_A = 0.77, theta_B = 0.55, ll = -30.76
Iteration: 4
theta_A = 0.78, theta_B = 0.53, ll = -30.33
Iteration: 5
theta_A = 0.79, theta_B = 0.53, ll = -30.07
Iteration: 6
theta_A = 0.79, theta_B = 0.52, ll = -29.95
Iteration: 7
theta_A = 0.80, theta_B = 0.52, ll = -29.90
Iteration: 8
theta_A = 0.80, theta_B = 0.52, ll = -29.88
Iteration: 9
theta_A = 0.80, theta_B = 0.52, ll = -29.87
```

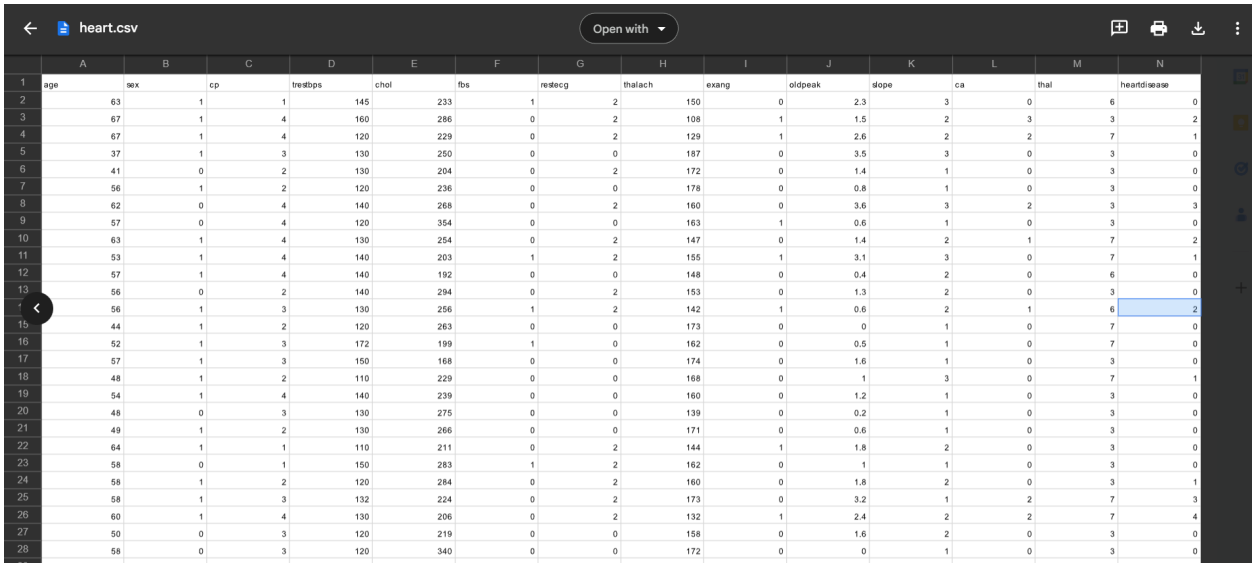
Out[1]: Text(0.5, 1.0, 'GMM Classification')



17.06.2023

Program 9: Write a program to construct a Bayesian Network considering training data. Use this model to make predictions.

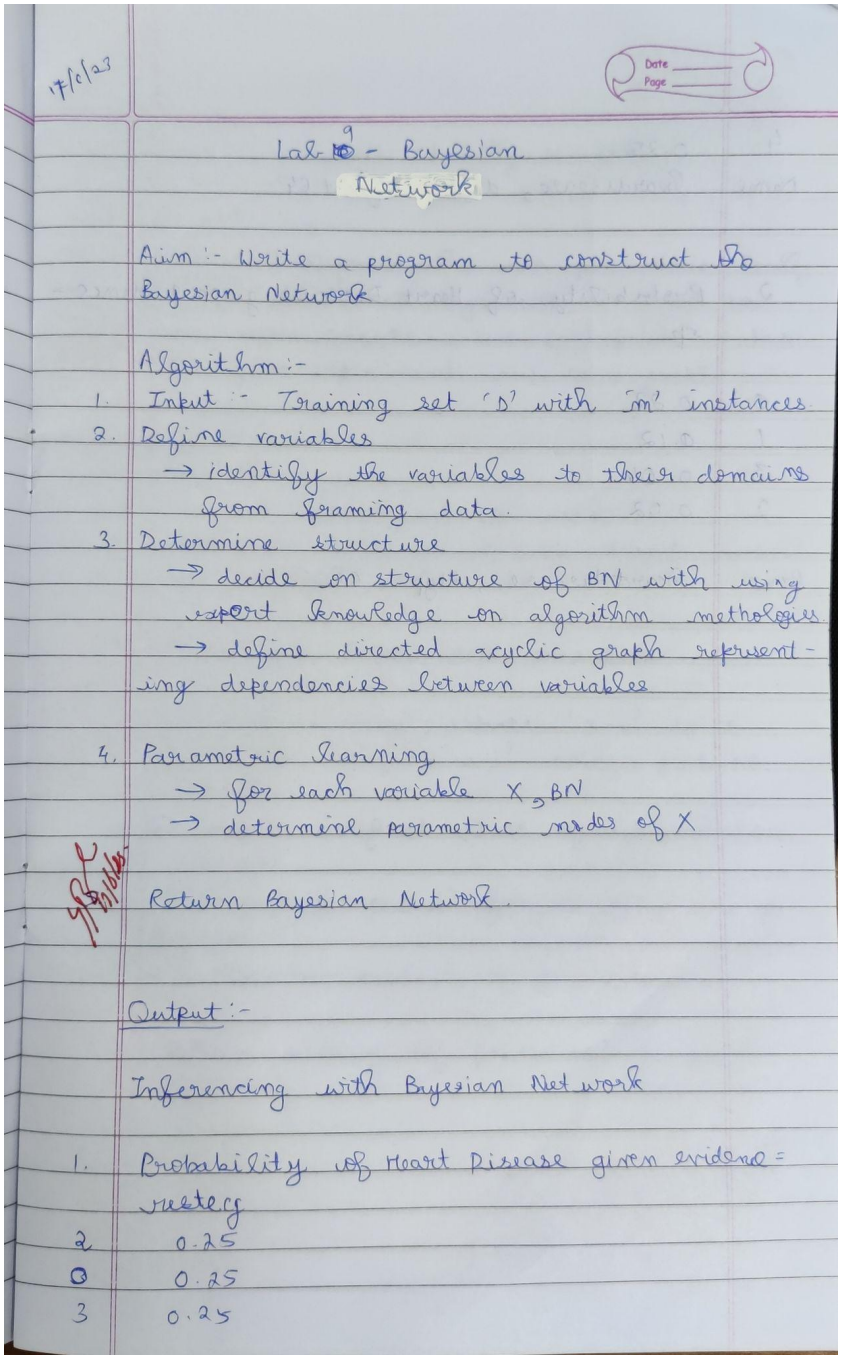
Dataset-heart


heart.csv

Open with

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
2	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
3	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
4	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
5	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
6	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
7	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
8	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
9	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
10	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
11	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
12	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
13	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
14	56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
15	44	1	2	120	263	0	0	173	0	0	1	0	7	0
16	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
17	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
18	48	1	2	110	229	0	0	168	0	1	3	0	7	1
19	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
20	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
21	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
22	64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
23	58	0	1	150	283	1	2	162	0	1	1	0	3	0
24	58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
25	58	1	3	132	224	0	2	173	0	3.2	1	2	7	3
26	60	1	4	130	206	0	2	132	1	2.4	2	2	7	4
27	50	0	3	120	219	0	0	158	0	1.6	2	0	3	0
28	58	0	3	120	340	0	0	172	0	0	1	0	3	0

Algorithm :



17/10/23

Lab 9 - Bayesian Network

Aim :- Write a program to construct a Bayesian Network

Algorithm :-

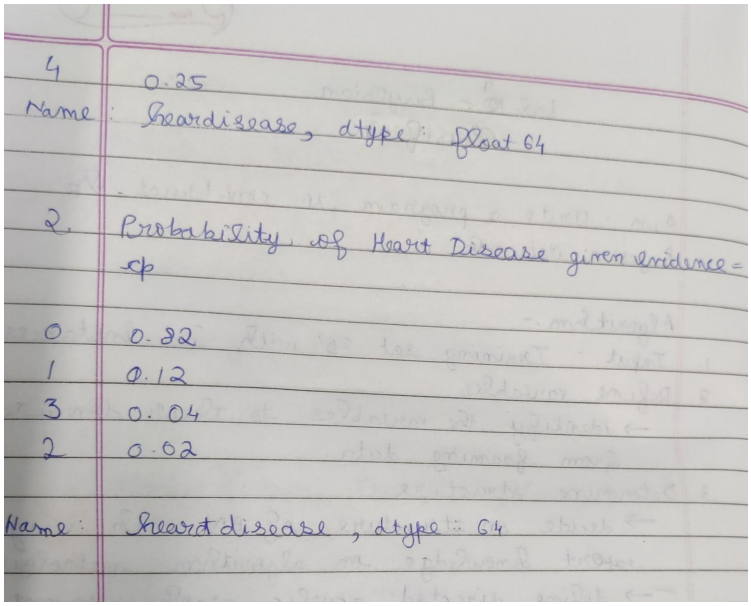
1. Input :- Training set 'D' with 'm' instances
2. Define variables
   → identify the variables to their domains from framing data.
3. Determine structure
   → decide on structure of BN with using expert knowledge on algorithm methologies
   → define directed acyclic graph representing dependencies between variables
4. Parametric learning
   → for each variable X, BN
   → determine parametric nodes of X

Return Bayesian Network.

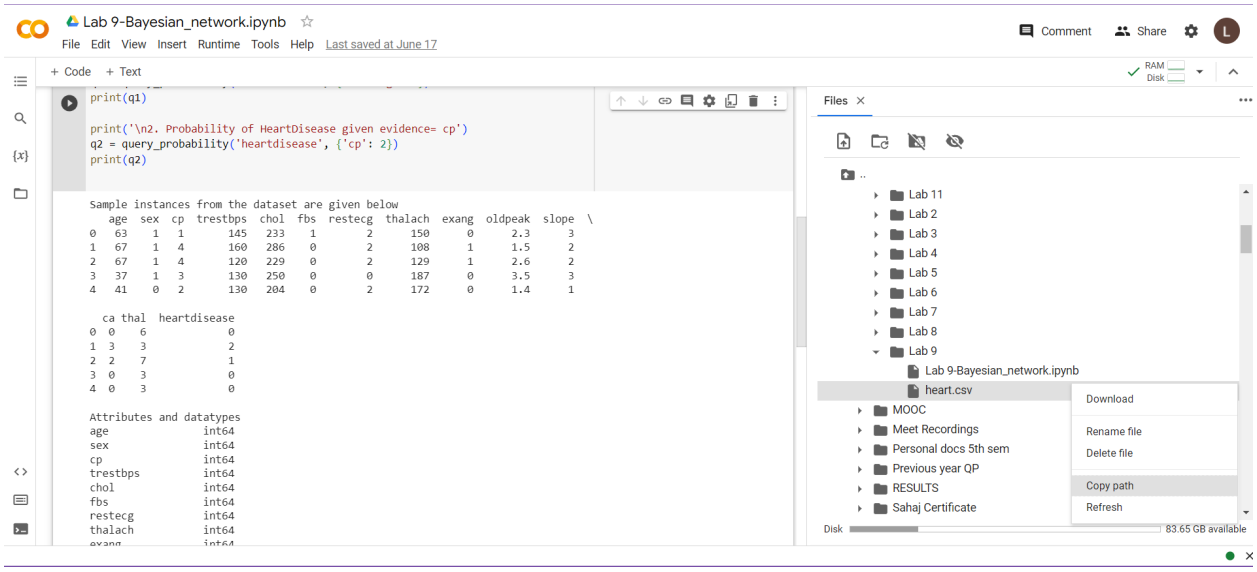
Output :-

Inferencing with Bayesian Network

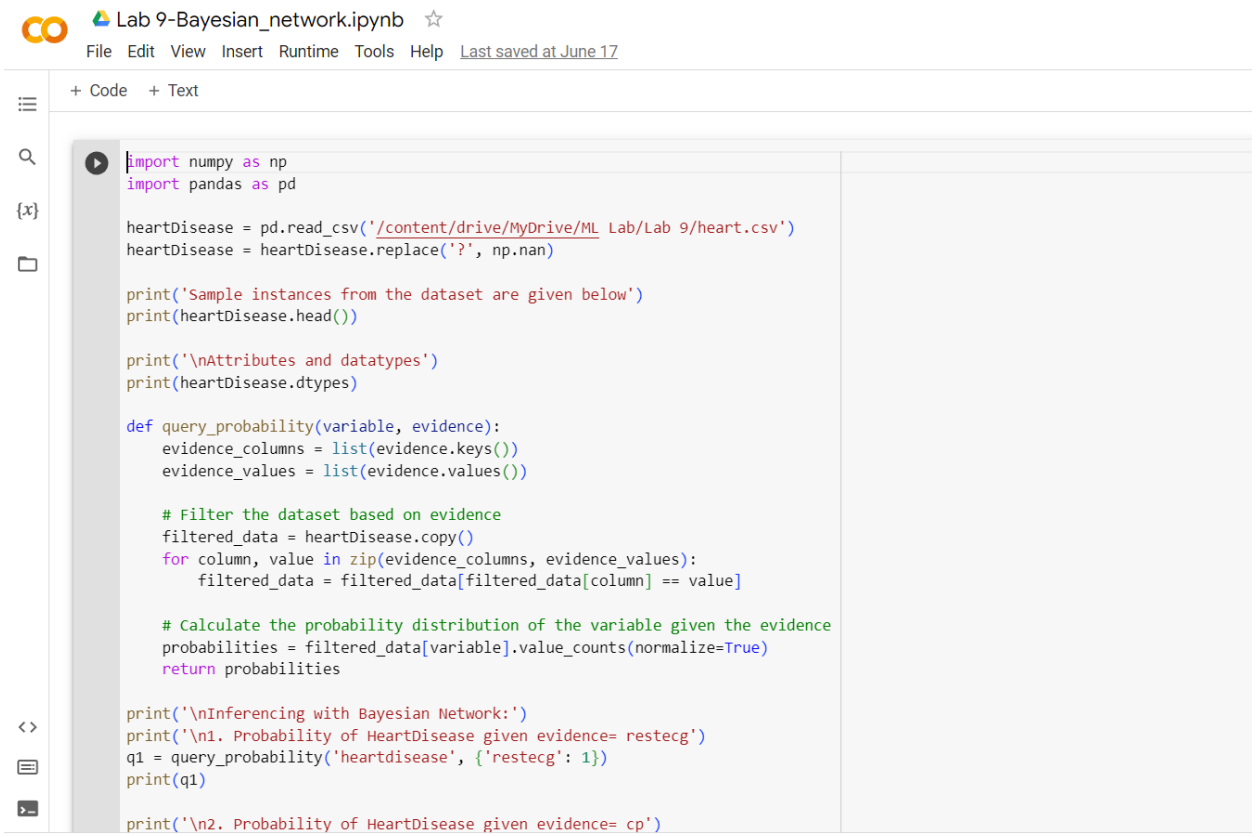
1. Probability of Heart Disease given evidence = restecg
2. 0.25
3. 0.25
3. 0.25



Uploading csv :



Program execution :



Lab 9-Bayesian_network.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at June 17

+ Code + Text

```
print(q1)

print('\n2. Probability of HeartDisease given evidence= cp')
q2 = query_probability('heartdisease', {'cp': 2})
print(q2)
```

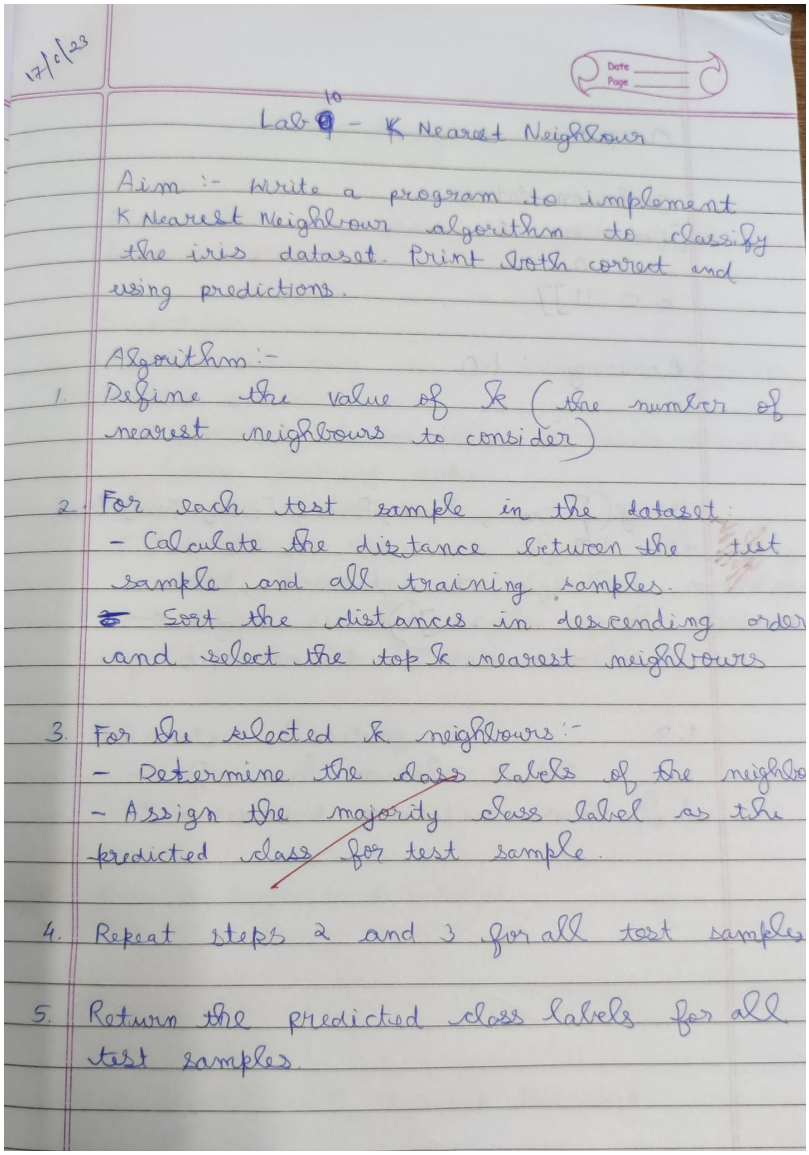
```
Sample instances from the dataset are given below
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   1       145   233    1        2      150      0      2.3      3
1   67    1   4       160   286    0        2      108      1      1.5      2
2   67    1   4       120   229    0        2      129      1      2.6      2
3   37    1   3       130   250    0        0      187      0      3.5      3
4   41    0   2       130   204    0        2      172      0      1.4      1

  ca thal  heartdisease
0  0    6             0
1  3    3             2
2  2    7             1
3  0    3             0
4  0    3             0

Attributes and datatypes
age               int64
sex               int64
cp                int64
trestbps          int64
chol              int64
fbs               int64
restecg           int64
thalach           int64
exang             int64
oldpeak         float64
slope             int64
ca               object
thal             object
heartdisease      int64
dtype: object

Inferencing with Bayesian Network:

1. Probability of HeartDisease given evidence= restecg
2    0.25
0    0.25
3    0.25
4    0.25
Name: heartdisease, dtype: float64

2. Probability of HeartDisease given evidence= cp
0    0.82
1    0.12
3    0.04
2    0.02
Name: heartdisease, dtype: float64
```

Date: 17.06.2023

Program 10– Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

Data set: Iris Dataset

Algorithm :



17/6/23

Lab 10 - K Nearest Neighbour

Aim :- Write a program to implement K Nearest Neighbour algorithm to classify the iris dataset. Print both correct and wrong predictions.

Algorithm :-

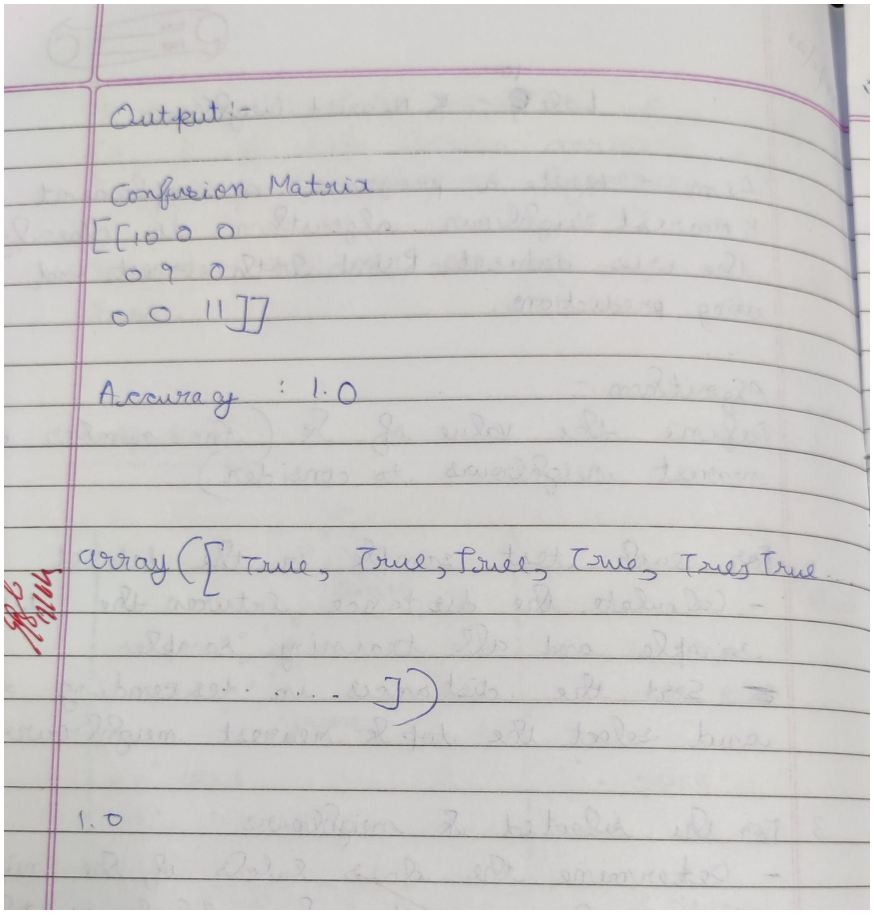
1. Define the value of K (the number of nearest neighbours to consider)

2. For each test sample in the dataset:
   - Calculate the distance between the test sample and all training samples.
   - Sort the distances in descending order and select the top K nearest neighbours

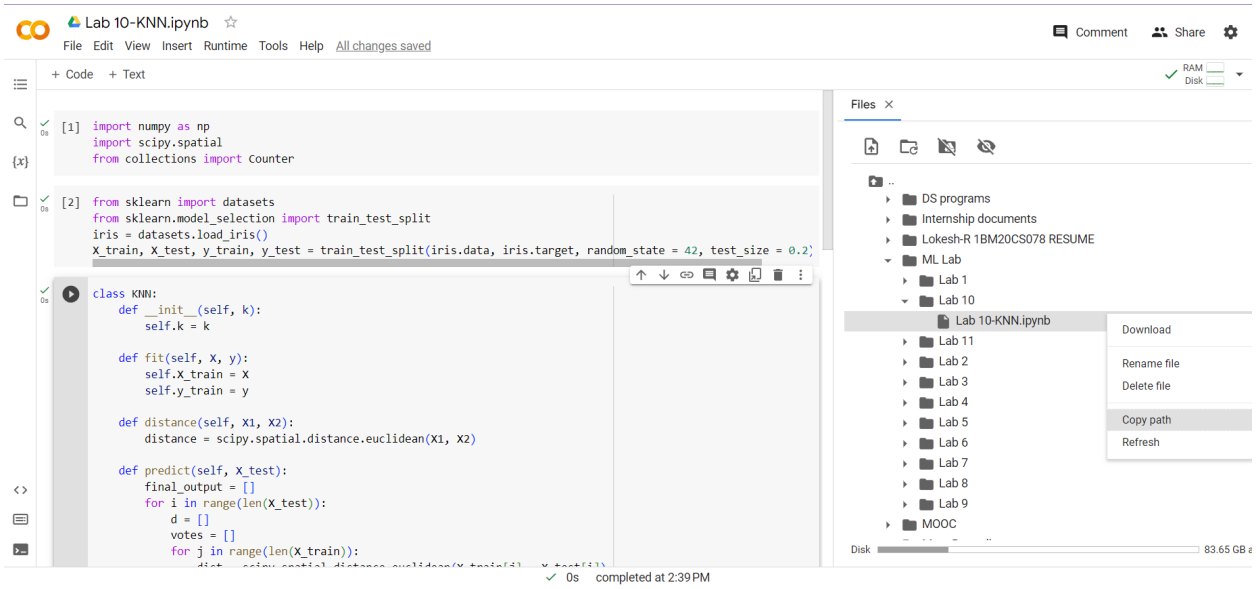
3. For the selected K neighbours :-
   - Determine the class labels of the neighbo
   - Assign the majority class label as the predicted class for test sample.

4. Repeat steps 2 and 3 for all test samples

5. Return the predicted class labels for all test samples.



Uploading csv :



Program execution :

Lab 10-KNN.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[1] import numpy as np
    import scipy.spatial
    from collections import Counter
```

```
[2] from sklearn import datasets
    from sklearn.model_selection import train_test_split
    iris = datasets.load_iris()
    X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, random_state = 42, test_size = 0.2)
```

```
class KNN:
    def __init__(self, k):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def distance(self, X1, X2):
        distance = scipy.spatial.distance.euclidean(X1, X2)

    def predict(self, X_test):
        final_output = []
        for i in range(len(X_test)):
            d = []
            votes = []
            for j in range(len(X_train)):
```

Lab 10-KNN.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
            for j in range(len(X_train)):
                dist = scipy.spatial.distance.euclidean(X_train[j] , X_test[i])
                d.append([dist, j])
            d.sort()
            d = d[0:self.k]
            for d, j in d:
                votes.append(y_train[j])
            ans = Counter(votes).most_common(1)[0][0]
            final_output.append(ans)

        return final_output

    def score(self, X_test, y_test):
        predictions = self.predict(X_test)
        return (predictions == y_test).sum() / len(y_test)
```

```
[4] clf = KNN(3)
    clf.fit(X_train, y_train)
    prediction = clf.predict(X_test)
    for i in prediction:
        print(i, end= ' ')
```

```
1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0
```

```
[5] prediction == y_test
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True])
```

```
[6] clf.score(X_test, y_test)
```

```
1.0
```

Date: 17.05.2023

Program 11:

Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Dataset :tips.csv

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2
14.78	3.23	Male	No	Sun	Dinner	2
10.27	1.71	Male	No	Sun	Dinner	2
35.26	5	Female	No	Sun	Dinner	4
15.42	1.57	Male	No	Sun	Dinner	2
18.43	3	Male	No	Sun	Dinner	4
14.83	3.02	Female	No	Sun	Dinner	2
21.58	3.92	Male	No	Sun	Dinner	2
10.33	1.67	Female	No	Sun	Dinner	3
16.29	3.71	Male	No	Sun	Dinner	3
16.97	3.5	Female	No	Sun	Dinner	3
20.65	3.35	Male	No	Sat	Dinner	3
17.92	4.08	Male	No	Sat	Dinner	2
20.29	2.75	Female	No	Sat	Dinner	2
15.77	2.23	Female	No	Sat	Dinner	2
39.42	7.58	Male	No	Sat	Dinner	4
19.82	3.18	Male	No	Sat	Dinner	2
17.81	2.34	Male	No	Sat	Dinner	4
13.37	2	Male	No	Sat	Dinner	2
12.69	2	Male	No	Sat	Dinner	2
21.7	4.3	Male	No	Sat	Dinner	2
19.65	3	Female	No	Sat	Dinner	2
9.55	1.45	Male	No	Sat	Dinner	2

Algorithm :

17/4/23

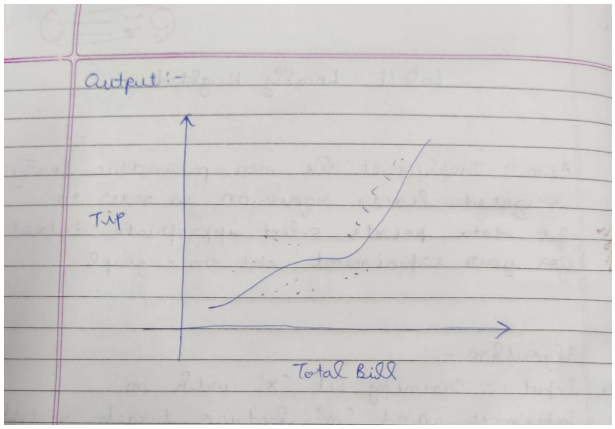
Lab-11 - Locally Weighted

Aim :- Implement the non parametric locally weighted linear regression in order to fit data points. Select appropriate dataset for your experiment and draw graph

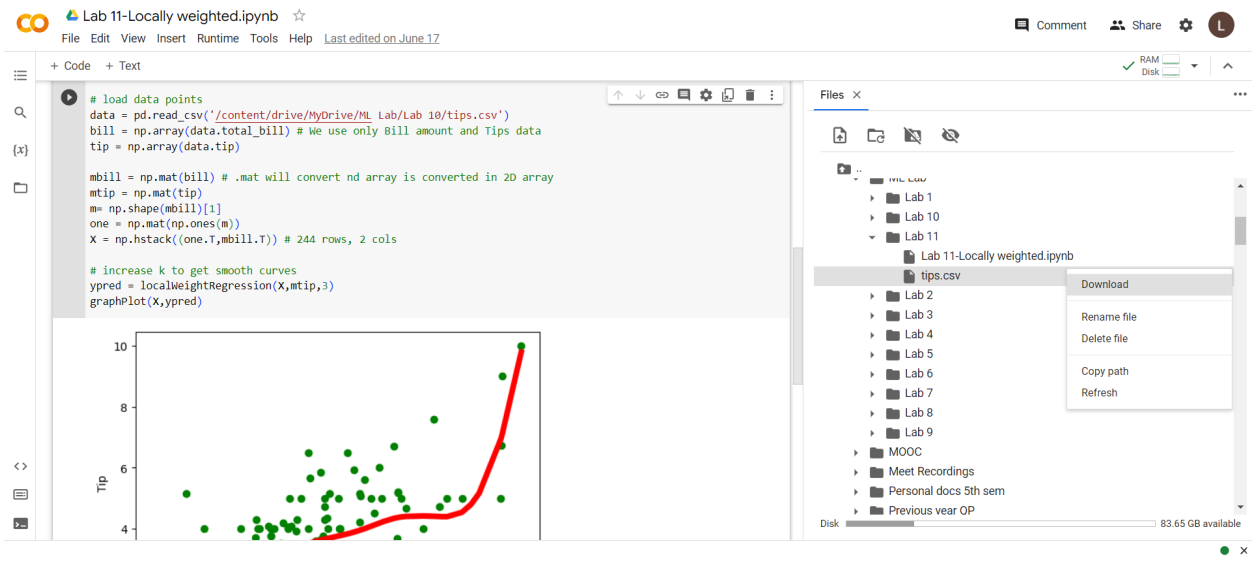
Algorithm :-

1. Input = Training set 'x' with 'm' instances and 'n' feature target variable 'y' instance
2. Initialise empty test prediction
3. For each training instance 'x-i' in x :
   a) Compute weight 'w-i' using gaussian kernel.
   b) Compute weighted least squares solution for x-i
   c) Predict 'y - pred_i' for 'x - query' using theta - i
   d) Add 'y - pred_i' to prediction.
4. Return average of 'prediction' as final prediction.

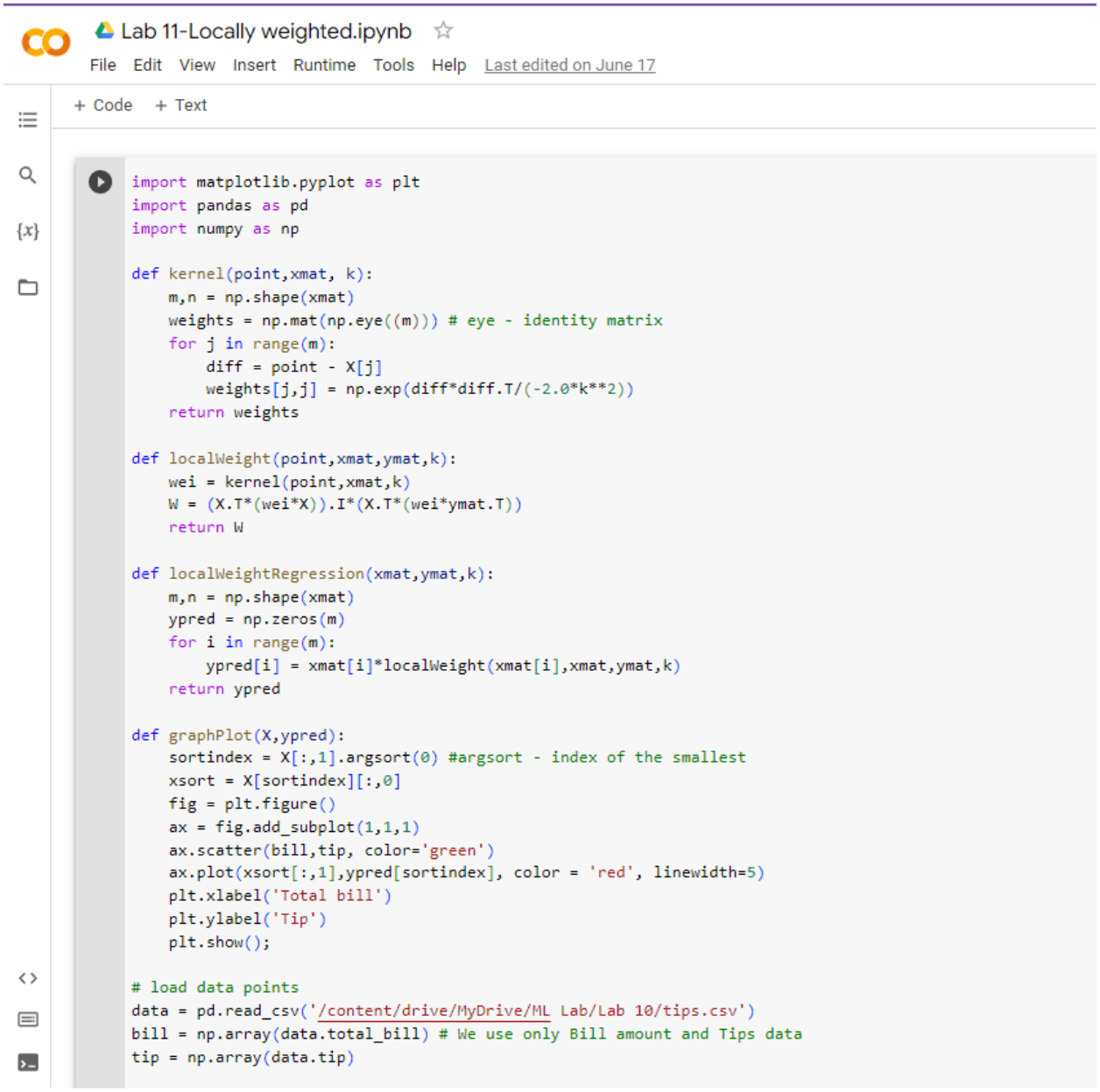
Output :-



Uploading csv :



Program execution



Lab 11-Locally weighted.ipynb

File Edit View Insert Runtime Tools Help Last edited on June 17

+ Code + Text

```
# load data points
data = pd.read_csv('/content/drive/MyDrive/ML Lab/Lab 10/tips.csv')
bill = np.array(data.total_bill) # We use only Bill amount and Tips data
tip = np.array(data.tip)

mbill = np.mat(bill) # .mat will convert nd array is converted in 2D array
mtip = np.mat(tip)
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T)) # 244 rows, 2 cols

# increase k to get smooth curves
ypred = localWeightRegression(X,mtip,3)
graphPlot(X,ypred)
```